ESD-TR-83-028

Semiannual Technical Summary
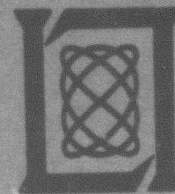
Distributed Sensor Networks

31 March 1983

Prepared for the Defense Advanced Research Projects Agency
under Electronic Systems Division Contract F19628-80-C-0002 by
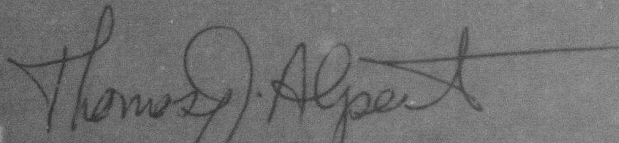
Lincoln Laboratory

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

LEXINGTON, MASSACHUSETTS

DTIC
ELECTE
S
OCT 0 4 1983
D
E

83 10 05 044

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER

Thomas J. Alpert, Major, USAF
Chief, ESD Lincoln Laboratory Project Office

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LINCOLN LABORATORY

DISTRIBUTED SENSOR NETWORKS

SEMIANNUAL TECHNICAL SUMMARY REPORT
TO THE
DEFENSE ADVANCED RESEARCH PROJECTS AGENCY

1 OCTOBER 1982 — 31 MARCH 1983

ISSUED 10 AUGUST 1983

LEXINGTON   MASSACHUSETTS

ABSTRACT

This report describes the work performed on the DARPA Distributed Sensor
Networks Program at Lincoln Laboratory during the period 1 October 1982
through 31 March 1983.

Accession For

NTIS GRA&I   X

DTIC TAB □

Unannounced □

Justification

By

Distribution/

Availability Codes

Avail and/or

Dist   Special

A

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

DISTRIBUTED SENSOR NETWORKS

I. INTRODUCTION AND SUMMARY

The Distributed Sensor Networks (DSN) program is aimed at developing and extending target surveillance and tracking technology in systems that employ multiple spatially distributed sensors and processing resources. Such a system would be made up of sensors, data bases, and processors distributed throughout an area and interconnected by an appropriate digital data communication system. Surveillance and tracking of low-flying aircraft has been selected to develop and evaluate DSN concepts in the light of a specific system problem. A DSN test bed that will make use of multiple small acoustic arrays as sensors for low-flying aircraft surveillance is being developed and will be used to test and demonstrate DSN techniques and technology. This Semiannual Technical Summary (SATS) reports results for the period 1 October 1982 though 31 March 1983.

Initial validation tests of a three-node message-based distributed acoustic surveillance system were conducted during this reporting period. The system consisted of three test-bed nodes interconnected and controlled by a separate experiment control and communication (ECC) computer. In addition to providing message-based internodal communications, the ECC also serves as a system user interface. Improvements needed to support detailed distributed system experiments were identified and have been partially implemented.

Signal-processing experiments have been conducted that indicate that it may be possible to significantly reduce the signal-processing computational load by reducing the number of microphones in each acoustic array, selecting analysis frequencies carefully, and generally making use of target characteristics. A new improved Maximum Likelihood Method of array processing was also tested. Initial results with that algorithm do not indicate any substantial improvement over the algorithms now in use in the test bed.

The data used for the signal-processing experiments were recorded from an earlier data acquisition experiment involving a single UH-1 helicopter as the acoustic source. Detailed plans have been formulated for a two-helicopter data acquisition experiment to provide data for future experimentation with multiple targets.

Also, a technique for simulating acoustic data for multiple targets by linearly combining real acoustic data from single targets was investigated and found to be practical.

A second multiple-target simulation method was also developed and used to investigate the behavior of test-bed azimuth tracking algorithms. In that approach, synthetic azimuth-time data are directly generated without simulating acoustic waveforms or front-end signal processing. This technique is a powerful tool for planning experiments with real aircraft and for investigating difficult experimental situations.

Algorithms for combining overlapping target position tracks resulting from track estimation by different test-bed node pairs were developed and initially tested during this reporting period.

Section III of this report summarizes progress that has been made in the development of standard multiprocessor communicating nodes for the DSN test bed. A prototype of such a node has been completed. The central element of the prototype is a multiple microcomputer system on a single bus. Separate processors perform tracking and communication functions. The prototype includes a custom-designed smart interface between the DSN test-bed node and a spread-spectrum radio unit that will be used for internodal communications and internodal ranging measurements. Protocols to provide broadcast communications and internodal ranging have been designed and are being implemented. The design includes the collection of communication performance statistics.

Section IV covers a number of items. These include the acquisition and restructuring of Columbia University-developed self-location software, procurement of a 68000 UNIX workstation to serve as a test-bed user interface computer, installation of ARPANET software, restructuring of signal-processing research tools and preparation of a paper that will be presented at the American Control Conference in June 1983.

II. DSN EXPERIMENTATION

Experimental DSN investigations conducted during this reporting period are described in the following sections.

A. NETWORK VALIDATION

Three of the test-bed nodes have been organized into an interim DSN test-bed configuration. Figure II-1 shows the three-node interconnection configuration while Fig. II-2 depicts the internal structure of a node. A PDP-11/70 computer is presently being used to implement the Experiment Control Computer (ECC) functions. Initial versions of user interface, system, communication, and application software have been completed. As a result of initial experimentation, desirable changes in the user interface were implemented. Necessary changes to internodal communication software and hardware were also identified and implemented.

Plans have now been formulated to validate the basic capabilities required for follow-on experimentation involving surveillance performance, data distribution within the test bed, and interprocess communications capacity. Validation experiments will be based on two data sets that will also be used in follow-on experiments. One data set will be comprised of peak data derived from a UH-1 helicopter flyby. The other data set will be comprised of synthetic peak data for a two-helicopter scenario. The synthetic data will be produced by the synthetic detection generator described in Sec. C.

The following paragraphs summarize (1) the status of the User Interface Program (UIP) and (2) the communication emulation changes which have been implemented or are under way.

The UIP is a program that resides upon the user computer and provides user services and an interface to the test bed. It now operates upon a PDP-11 system running UNIX. It will subsequently be installed on a 68000 system running UNIX and will become the test-bed ECC processor. The UIP presently allows the user to:

(1) Start, stop, and interrogate processes in the nodal CPUs.
(2) Modify the nodal tracking parameters.
(3) Examine the memories of remote processors while they are running.
(4) Use prepared command files to run complex experiments.
(5) Perform breakpoint debugging of processes in the nodal processors.

In addition the UIP will:

(6) Provide broadcast communication emulation by means of land lines.
(7) Spool tracking messages to disk for subsequent analysis.
(8) Log all command, reply, and error messages into a disk file for analysis.
(9) Inject track messages into the network from files and thereby simulate extra nodes.
(10) Provide console input/output services for remote processes.

In addition, we have made improvements in the azimuth and position display processes which run on the PDP-11/70 as adjunct processes to the UIP. These processes plot tracks
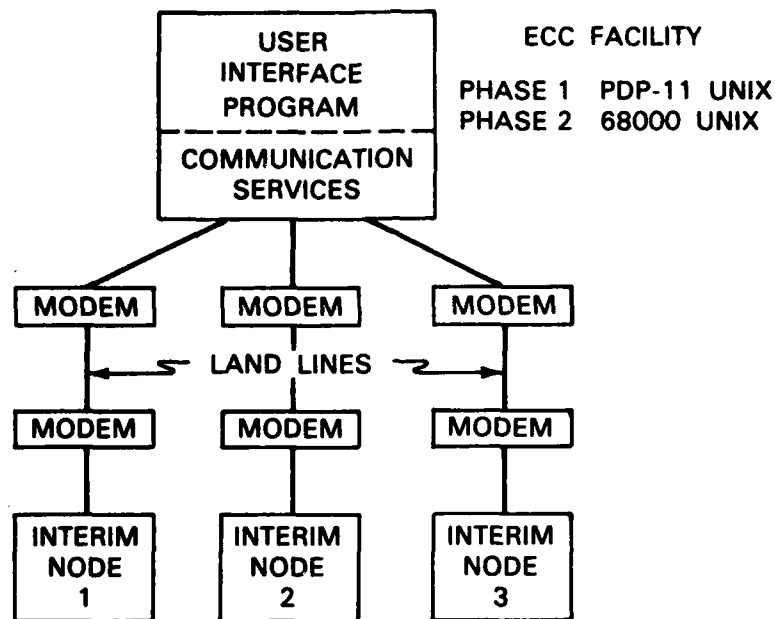
3

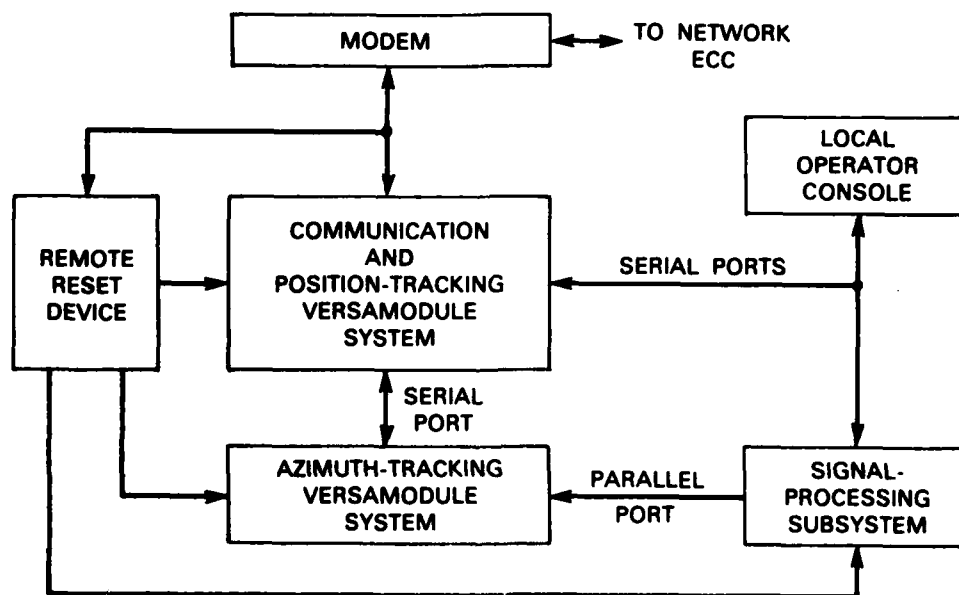Fig. II-1. Interim three-node test-bed configuration.

Fig. II- Interim tr   ad node configuration.

4

from one or more nodes. They take their input from the UNIX buffer pool as the azimuth and position track messages are spooled to disk. These processes allow the generation of tracks to be monitored in real time during an experiment.

A number of changes were required to improve the performance of the communication emulation function operating on the PDP-11. A multichannel DMA-output, FIFO-input serial line controller was installed. This controller is used for all communications between the PDP-11 and the nodes. It supports one full duplex 9600 baud line to each node. A series of experiments was performed to determine the best method of using this controller with the UNIX operating system to obtain the highest message throughput rate. Line-at-a-time receptions combined with some kernel additions gave the highest throughput; a little over three messages per link, per direction, per second on three links simultaneously.

Rather than being a separate process, the communications emulation feature is part of the UIP, so that the PDP-11/70 will be efficiently utilized and message traffic can be easily logged onto disk. Necessary performance improvements required considerable tuning of the UIP's internal tasking and message routing code to make it run as fast as possible and to avoid network-wide deadlock situations. In conjunction with this, we have implemented the capability to slow down the data input to the tracking process. This will allow experimentation with situations with more nodes (real or simulated) and more message traffic than would be possible in real time.

We are in the process of adding the capability of limiting the message traffic generated by each node, and of selectively enabling and disabling communications links between nodes.

B. SIGNAL PROCESSING

Progress in the area of signal-processing experimentation is summarized in this section. Experiments have been conducted to measure the performance of signal-processing algorithms. A methodology for using single target data to simulate multiple target data has been investigated and appears to be a promising experimental technique. Detailed plans have been formulatd for multiple-target data acquisition experiments.

1. Wavenumber Analysis with Fewer Sensors

The performance of our current DSN signal-processing system has been measured as a function of the number of sensors used for analysis. We reduced the number of analysis channels by using subsets of the nine-sensor DSN array. The data set used for this analysis consisted of UH-1 helicopter data collected in earlier field experiments. We found that with just three-sensors we could get performance comparable to that with the full nine sensors, provided the three-sensor subset is chosen in accordance with the expected frequency characteristics of the target.

The test-bed microphone configuration for these experiments is illustrated in Fig. II-3. The microphones are located on three equilateral triangles with sides of 6, 2, and 0.75 m. The signal-processing algorithms in the test-bed nodes operate upon all nine channels of data. First the frequencies are selected that correspond to power peaks in the temporal spectra of the data. The eight frequencies with the largest energy are saved. Most of the spectral energy and therefore the selected frequencies lie between 12 and 100 Hz for the UH-1

5

Fig. II-3. Nine-sensor DSN array.
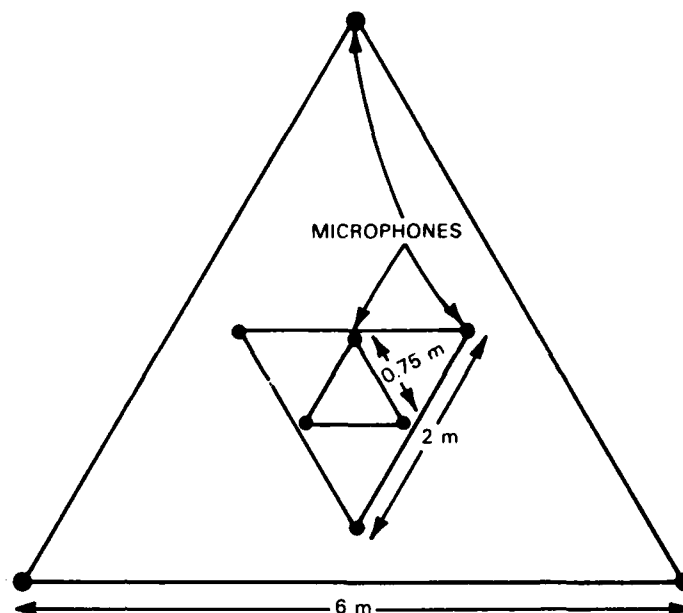
data (see Fig. II-4). Power spectral density matrices are then estimated for each of the eight selected frequencies. A MLM wavenumber spectrum is computed at each of the frequencies. This spectrum is evaluated for 120 uniformly separated azimuths from 0° to 360° and eight elevations from 0° to 80°. An algorithm to extract power peaks in elevation-azimuth space is the final signal-processing step.

Figures II-5 illustrate the results obtained when the number of sensors was reduced from 9 to 3. Each of the plots represents power peaks as a function of their azimuth location and time of detection. For each analysis interval, the highest power peak and all other peaks within 10 dB of that peak are shown. Part (a) of the figure is a plot obtained by analyzing UH-1 data with all nine sensors using the standard DSN signal-processing procedure outlined above. Figure II-5(b) is a plot obtained for the same data using the standard DSN signal-processing procedure except that only the three outermost sensors were used. This plot contains "ghost" contributions that are reflections of the helicopter tracks at different azimuths. This effect takes place due to spatial aliasing. This is to be expected since the 6-m distance between microphones is more than half a wavelength of sound for frequencies above 30 Hz. From Fig. II-4, we note that much of the UH-1 spectral energy is above the 30-Hz limit. To counter this problem, one approach is to decrease the distance between the sensors. Figure II-5(c) represents such a case, where the three innermost sensors have been used. However, since the array aperture has decreased, poorer resolution of the helicopter track can be seen in the form of a wider azimuth band of detection. A different solution to the problem of aliasing is to use the outermost sensors but to restrict the frequencies of analysis to below 30 Hz. This analysis results in the plot of Fig. II-5(d). Clearly, there is no aliasing and the peaks are quite sharp from a resolution point of view.

6
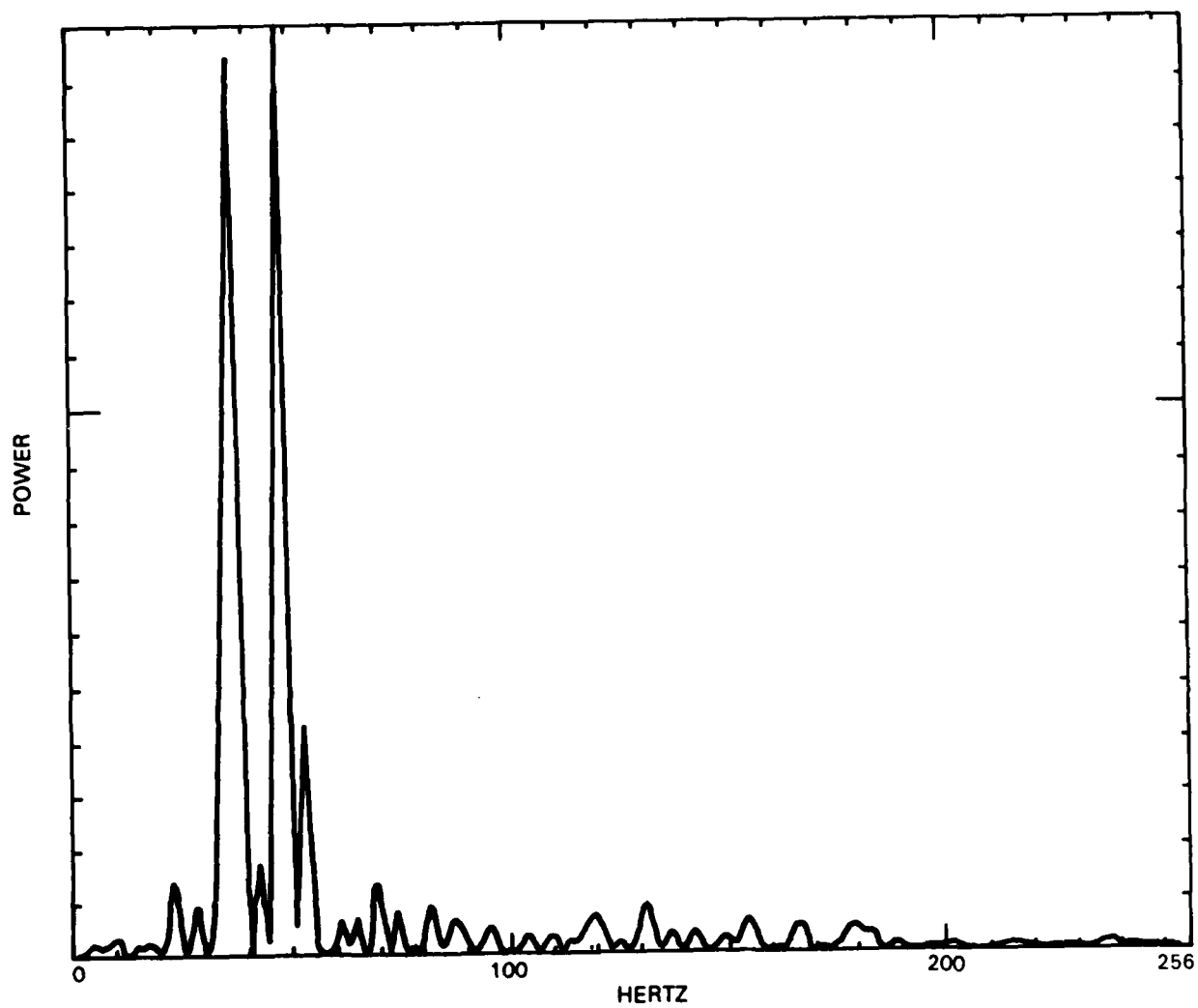
Fig. II-4. Spectral content of UH-1 acoustic data.

7
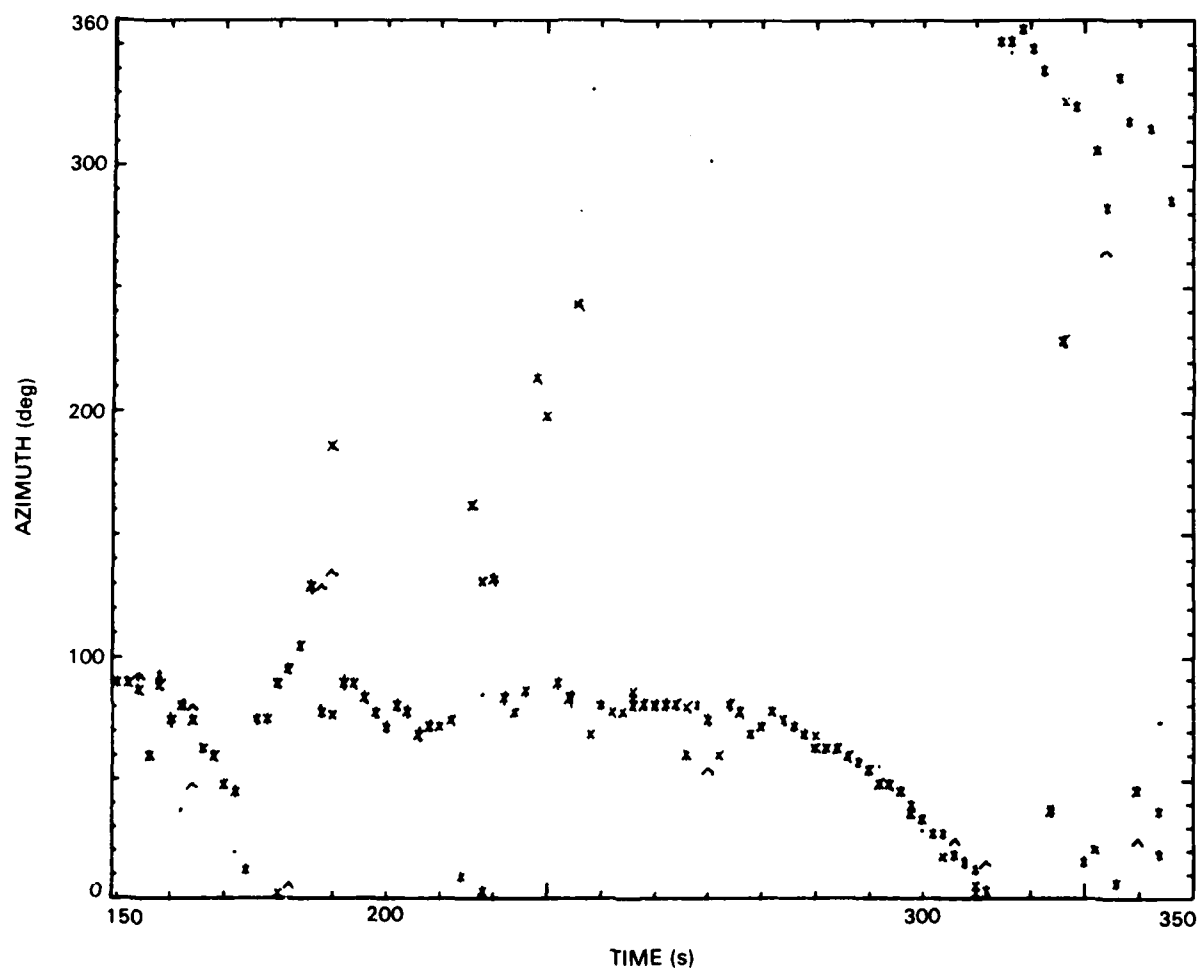
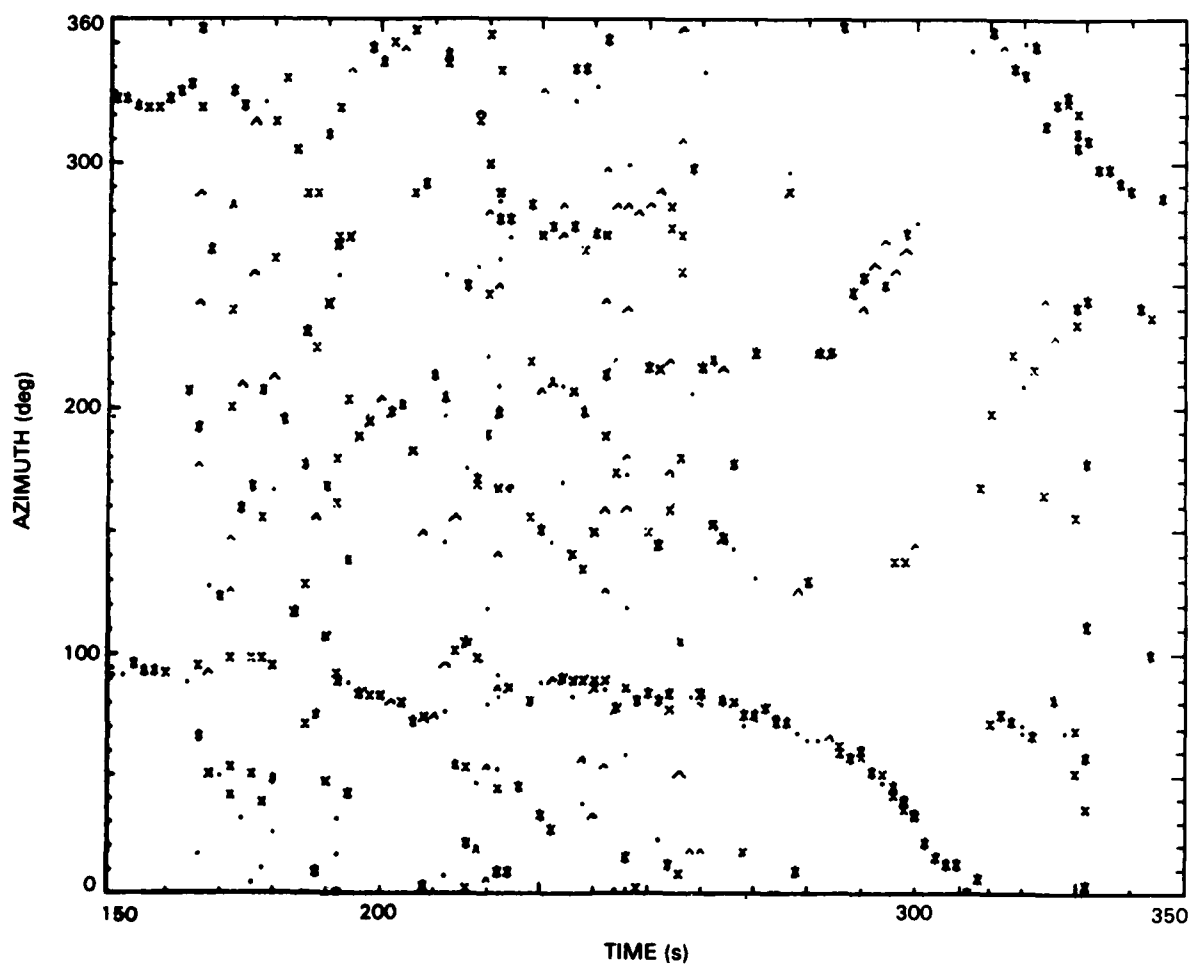Fig. II-5(a). Standard DSN nine-channel processing.

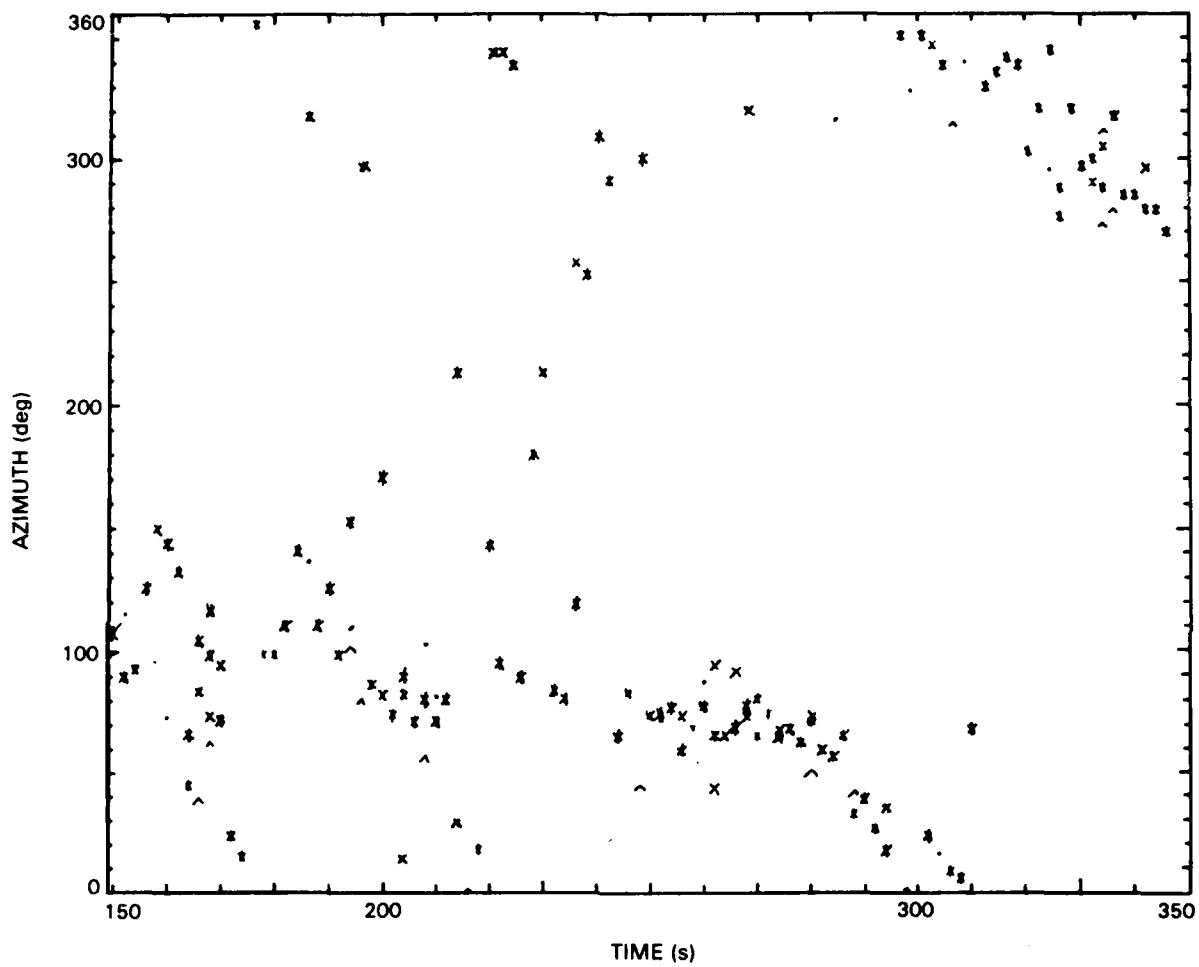Fig. II-5(b). Standard DSN processing with three outermost channels.

9

Fig. II-5(c). Standard DSN processing with three innermost channels.
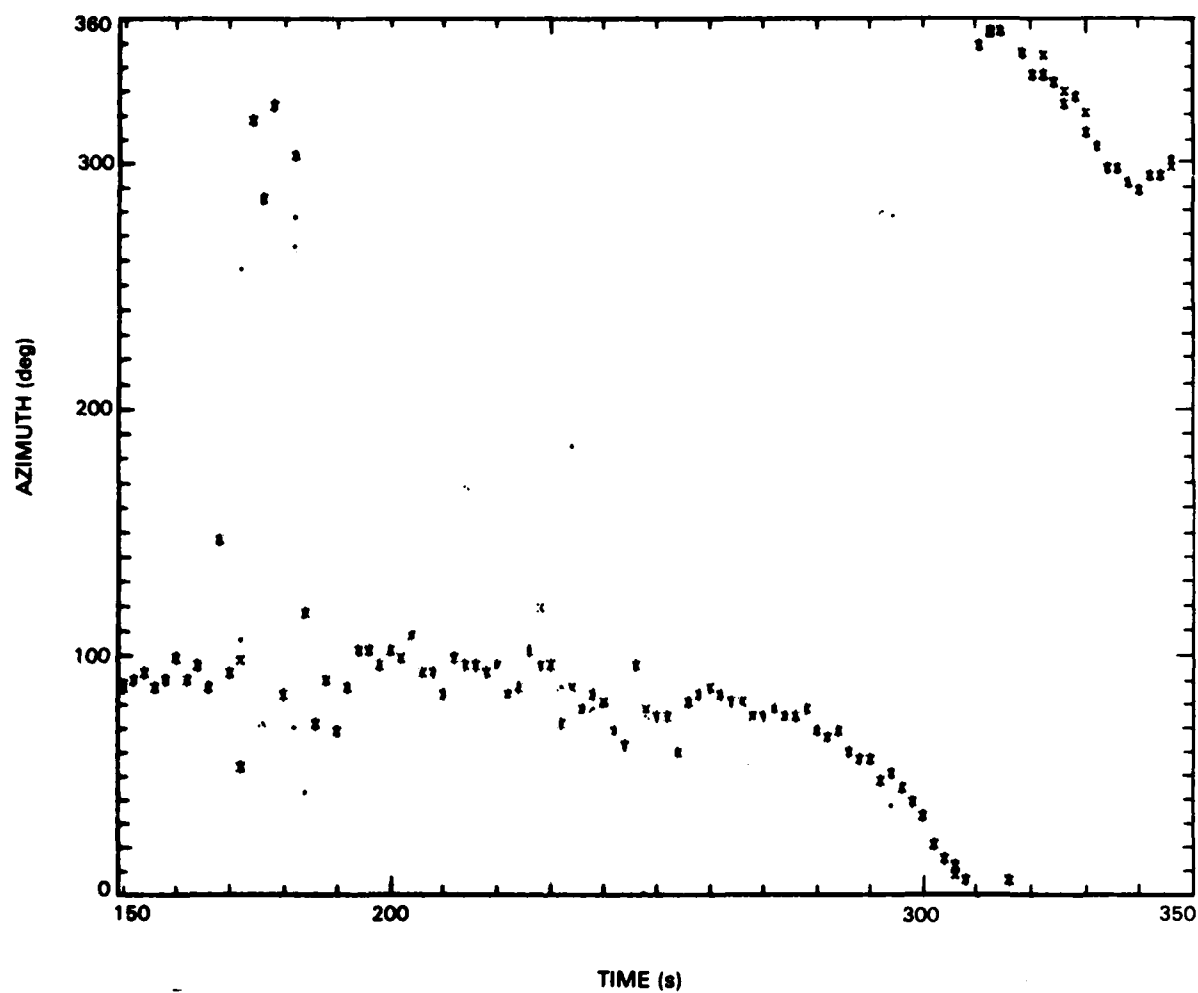
129687-R

10

Fig. II-5(d). Standard DSN processing with three outermost channels restricted to 10 to 30 Hz.
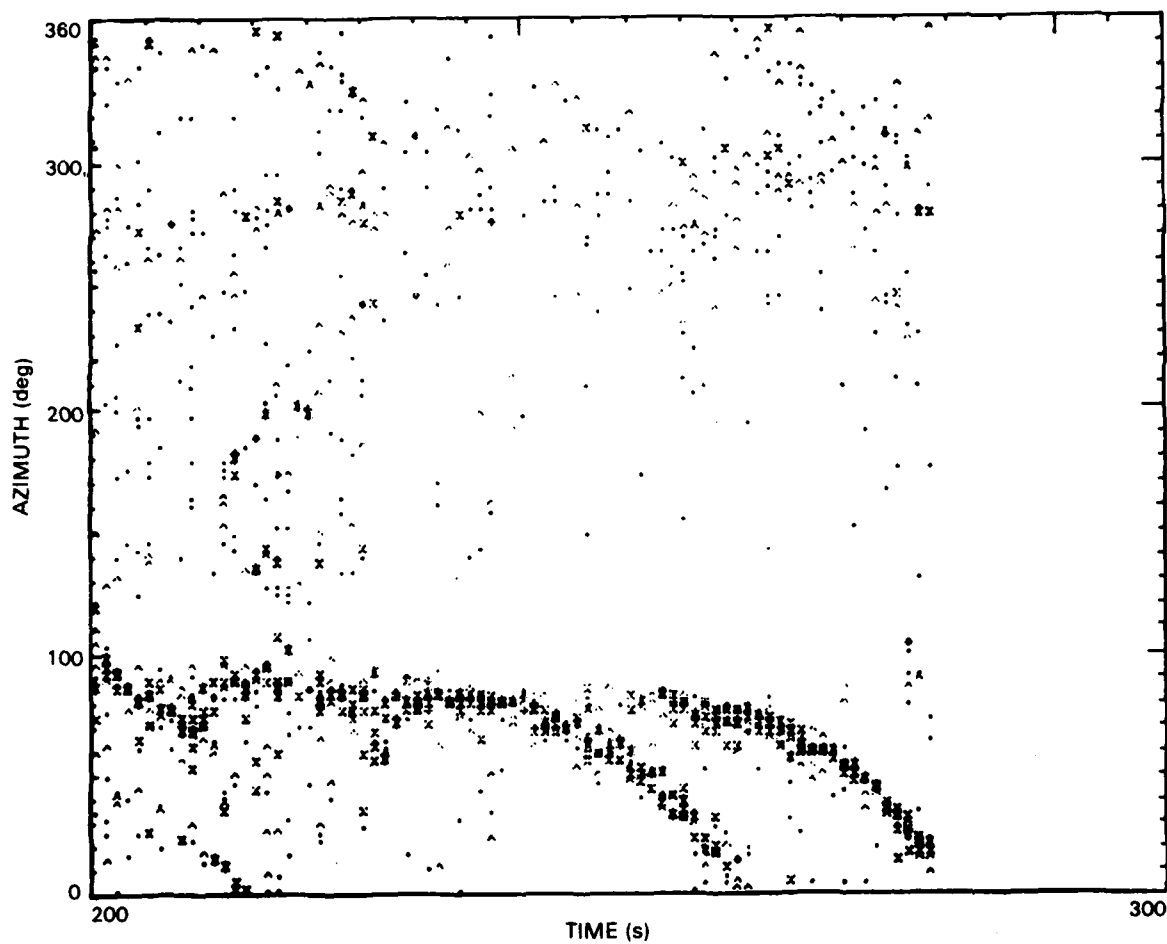
Fig. II-6. Standard DSN processing for simulated trailing UH-1 helicopter data.

2. Performance of a New Maximum Likelihood Method

The performance of a proposed improved MLM procedure* has been investigated. This method estimates the wavenumber spectrum using the formula

$$1/ \{[1/MLM(N)] - [1/MLM(M)]\}$$

where MLM(N) is an MLM spectrum of order N and N > M. The resulting spectrum is guaranteed to be positive since a lower-order MLM spectrum is always greater than a coresponding higher-order MLM spectrum. The key to the "improvement" in this method lies in the fact that for narrowband signals MLM(N) is closer to MLM(M) than it is for a more broadband signal. The maximum difference between the two takes place for white noise. The new procedure produces a large response when MLM(N) is close to MLM(M) (i.e., for narrowband signals) and produces a value close to MLM(N) when MLM(N) is much smaller than MLM(M) (i.e., for broadband signals).

This formula was applied to the UH-1 data using MLM wavenumber spectra produced by the current DSN algorithm. No significant performance improvement was noted. We have hypothesized two possible reasons for this lack of improvement. One hypothesis is that the covariance estimates from the UH-1 data do not correspond to sufficiently narrowband MLM spatial spectra. (The spectral analysis is performed with only 4-Hz resolution.) The second hypothesis is that the DSN sensor positions are not suitable for the new procedure. The originators of the technique used uniform rectangular sensor grids in their experiments. We are currently testing both these hypotheses.

3. Multitarget Data Simulations Using Single-Target Data

For studying the performance of signal-processing algorithms in the presence of multiple targets, it is necessary to study a number of situations for which real data are very difficult or even impossible to obtain. We have designed a simulation strategy in which real single-target data are used to simulate multiple-target situations. The strategy is to combine weighted and delayed versions of real single-target data. As an example of such a simulation, we combined UH-1 data collected at one of the DSN nodes with a version of itself delayed by 50 s. Obviously, this is intended to simulate a situation where one UH-1 helicopter is trailing another, separated in time by 50 s. When we analyzed this simulated waveform data, we obtained a peak file illustrated in the azimuth-time plot of Fig. II-6. The characteristics of this peak file appear reasonable.

4. Multitarget Experiment Plan

In order to validate our multiple-target data simulations as well as to obtain real multitarget data, we have designed a field experiment in which two helicopters will be used. The field experiment, to be carried out at Hanscom AFB, has been designed to collect data which we expect will be interesting from both the signal-processing and tracking points of

---

*J.S. Lim and F.U. Dowla, "Improved Maximum Likelihood Method for Two-Dimensional Spectral Estimation," Proc. 1983 Intl. Conf. on Acoustics, Speech and Signal Processing, Boston, April 1983, pp. 851-855.
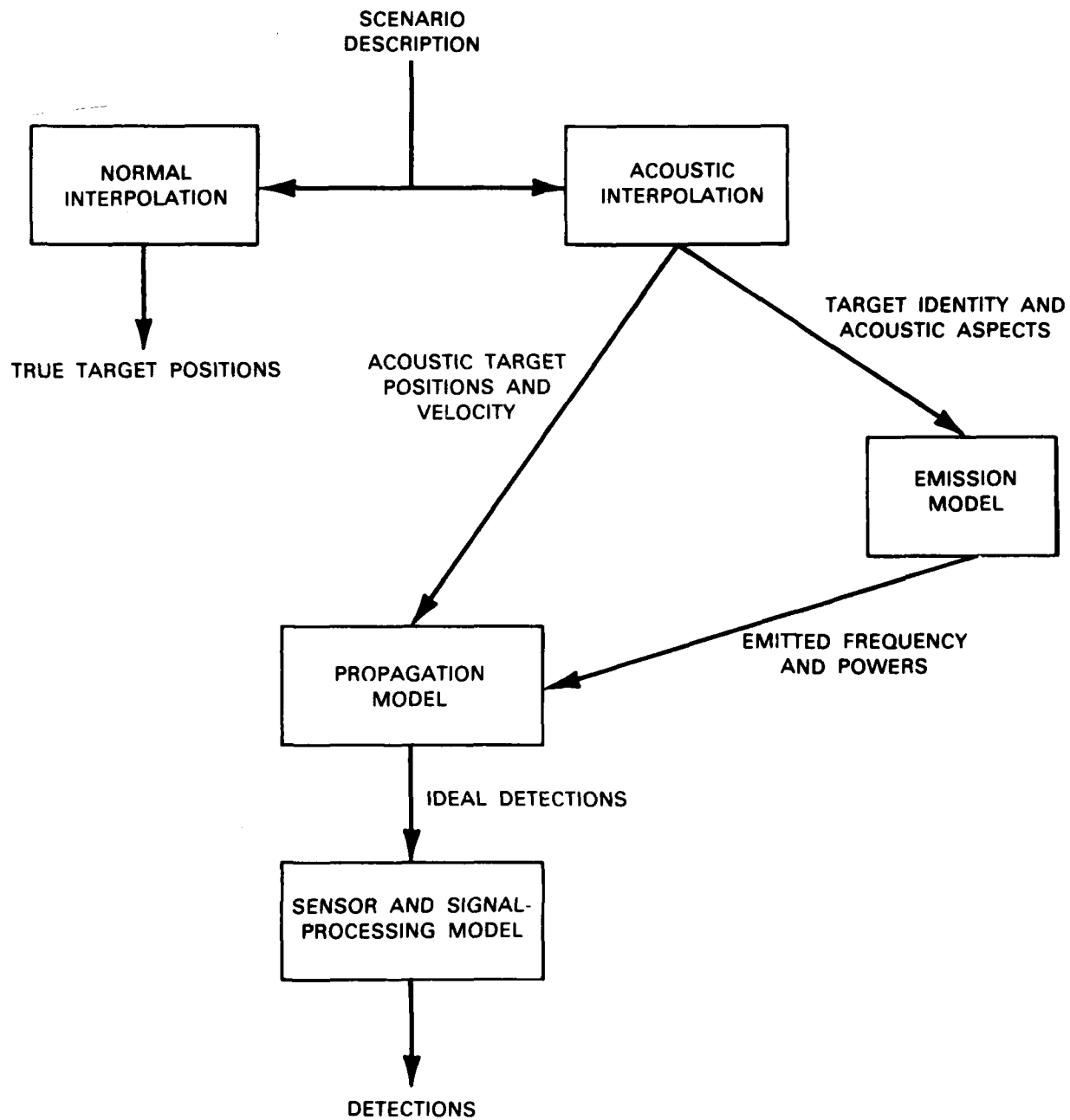
Fig. II-7. Data flow between models in synthetic detection generator.

view. In particular, we will be collecting data for scenarios in which one helicopter is trailing another along a straight path parallel to the DSN nodes. In another scenario, the two helicopters will be traversing the same straight path but they will be coming from opposite directions at different elevations. The experiment will also include data collection from hovering helicopters under various conditions.

C. SYNTHETIC DETECTION DATA

A synthetic detection generator has been developed to simulate the output from nodal signal-processing systems. This generator is being used to experiment with tracking multiple targets. The synthetic waveform generator (see Sec. B) provides a related capability but at the detailed level of the actual signal waveforms. The synthetic detection generator provides for experimentation with tracking and higher-level functions without the need to develop and execute all the lower-level signal-processing functions. The synthetic detection generator does not create synthetic waveform data.

1. Design and Testing of the Synthetic Detection Generator

Figure II-7 illustrates the major data flows in the synthetic detection generator. A scenario is described in terms of target identities and trajectories. The trajectories are described by piecewise continuous straight line segments with either constant target velocity or acceleration along each segment.

Target positions are interpolated in two ways. True target positions and velocities are interpolated from the trajectory descriptions at each sensor sample time. The resulting data provide a "ground truth" against which the performance of tracking algorithms can be measured. Acoustic target positions and velocities relative to a particular sensor are also calculated at each sensor sample time. A target's acoustic position at a particular sensor sample time is the position it had at the time it emitted the sound received by the sensor at the sample time. Acoustic velocity is similarly defined.

The acoustic interpolation outputs are provided to an emission model and to a propagation model. The emission model generates targets using one or more tones of constant frequency. The power level can be aspect-angle dependent. The propagation model introduces Doppler shift and reduces the power levels by the square of the sensor to target acoustic range.

In the figure, the propagation model output is labeled "ideal detections." If the sensor and signal processor were ideal, the signal processor would produce ideal detections. The sensor and signal-processor model modifies and/or discards the elements of the list of frequencies and power levels that constitute the ideal detections to produce the final synthetic detection data.

Figure II-8 illustrates the sensor and signal-processor model. Random measurement errors are added to ideal detections and false alarms are introduced. The measurement error module also discards data on the basis of signal-to-noise ratios. The frequency processing module limits the number of frequencies and the band of frequencies. The resolution module combines detections which are not resolvable in frequency, azimuth angle, and elevation angle.

IDEAL DETECTIONS

```
┌─────────────┐                    ┌─────────────┐
│   FALSE     │                    │ MEASUREMENT │
│   ALARM     │                    │   ERROR     │
│  SUBMODEL   │                    │  SUBMODEL   │
└─────────────┘                    └─────────────┘
```

FALSE   ERRORFUL
DETECTIONS   DETECTIONS

```
              ┌─────────────┐
              │  FREQUENCY  │
              │ PROCESSING  │
              │  SUBMODEL   │
              └─────────────┘
```

POTENTIAL
DETECTIONS

```
              ┌─────────────┐
              │ RESOLUTION  │
              │  SUBMODEL   │
              └─────────────┘
```
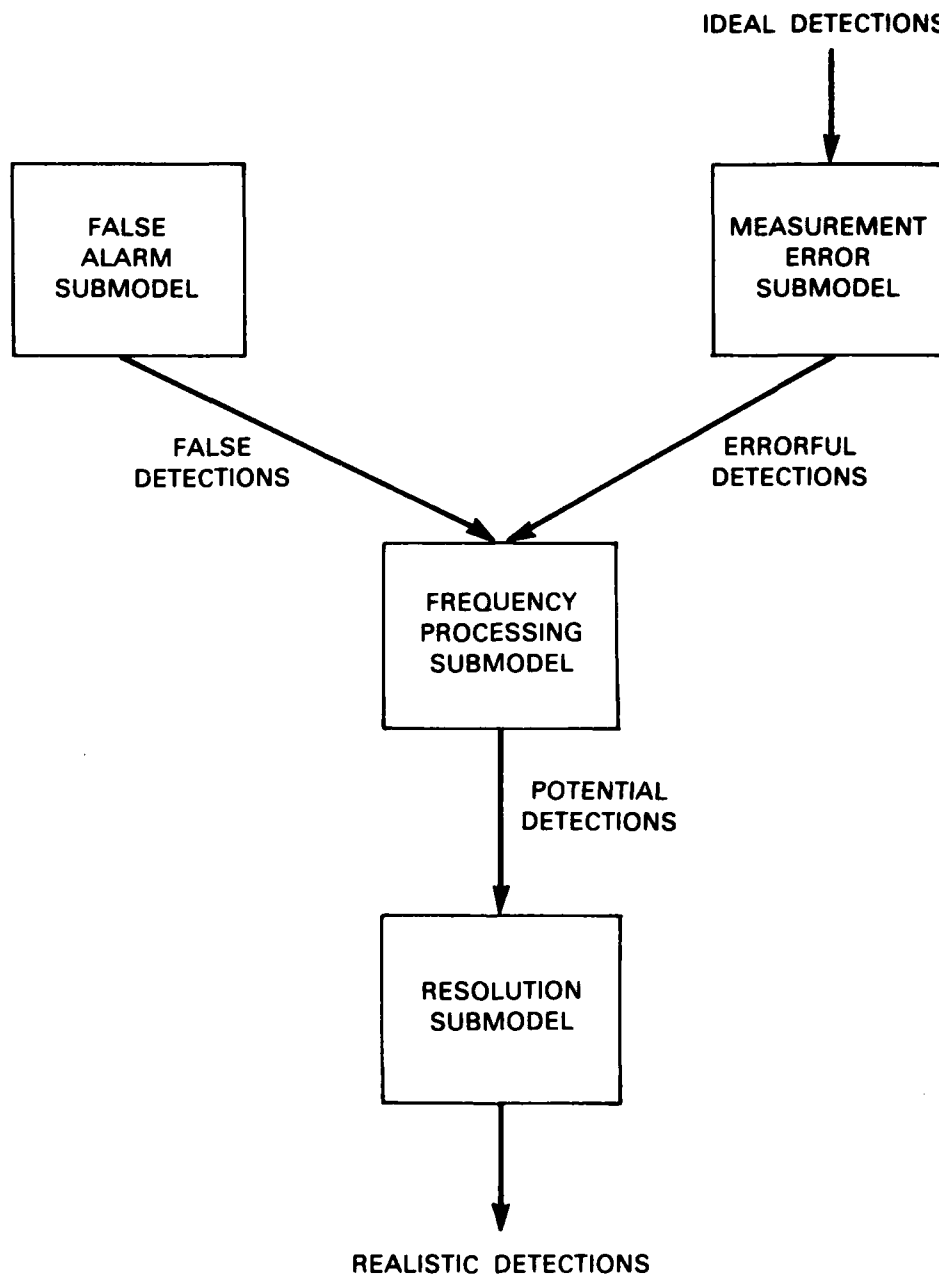
REALISTIC DETECTIONS

Fig. II-8. Data flow between submodels in sensor and signal-processor model.

129691-N

16

The different models are quite simple. However, the organization of the synthetic detection generator as a collection of interconnected models allows easy substitution of more sophisticated models as desired. However, the need for more sophisticated models is not clear. It depends upon how well the overall system mimics real data.

Initial model validation tests with helicopter data have indicated that the model produces output quite similar to that obtained by processing real acoustic data. Figures II-9 and -10 show actual signal-processor output data and synthetic generator data for the flyby of a UH-1 helicopter. The plots should not be expected to match perfectly since the helicopter trajectory during the experiment is only known approximately and not enough information is available to allow secondary noise sources to be included in the synthetic generator. The most notable difference is the loss of detection of the helicopter after about 370 s in Fig. II-9. This may be due to terrain masking or other propagation phenomena that are not included in the propagation model of the synthetic generator. Overall, the differences are modest compared to the degree of agreement in the target detections during most of the time interval.

2. An Experiment Using the Synthetic Detection Generator

An initial set of multiple-target data has been produced using the synthetic detection generator. These data have been used for initial experimentation with signal processing and azimuth tracker performance. They provide more information concerning the "capture" effect of our signal-processing algorithms in which a loud sound does not allow a quieter one to be detected and more information on how the azimuth tracker can be confused by multiple targets.

The experimental scenario was quite simple. Figure II-11 illustrates test-bed node locations and a UH-1 helicopter trajectory for a previously executed data-acquisition experiment. The synthetic detection generator was used to produce detection data for two helicopters flying along the same trajectory, one from east to west and the other from west to east. The two helicopters were simulated as flying at the same velocity and crossing at their point of closest approach to the node H sensor. Five minutes of data were generated with the crossing time occurring in the middle of the run. At the beginning and end of that interval, each sensor "sees" essentially the same situation: two targets, with effectively identical spectral shapes (Doppler shift included) and similar power levels, roughly 180° apart.

For the node H sensor, the near identity of spectral shapes and the similarity of power levels persists as the targets move toward each other and cross. The symmetry of the patterns of detections in Fig. II-12 illustrates this point. Overlaid on the detections are lines showing azimuth tracks. The tracker does well except for the 20 s bracketing the crossing time.

More extreme gaps can be found in Fig. II-13, the plot for node L. In this case, the tracker treats the data as if there is one target which flies from east to west and back, plus an intermittent target in the far west. This is an example of signal processor "capture." The signal processor in each node can only process eight frequency channels in each processing interval and can be "captured" if one target's sound emissions dominate and cause all eight frequency channels to be used for that one target.
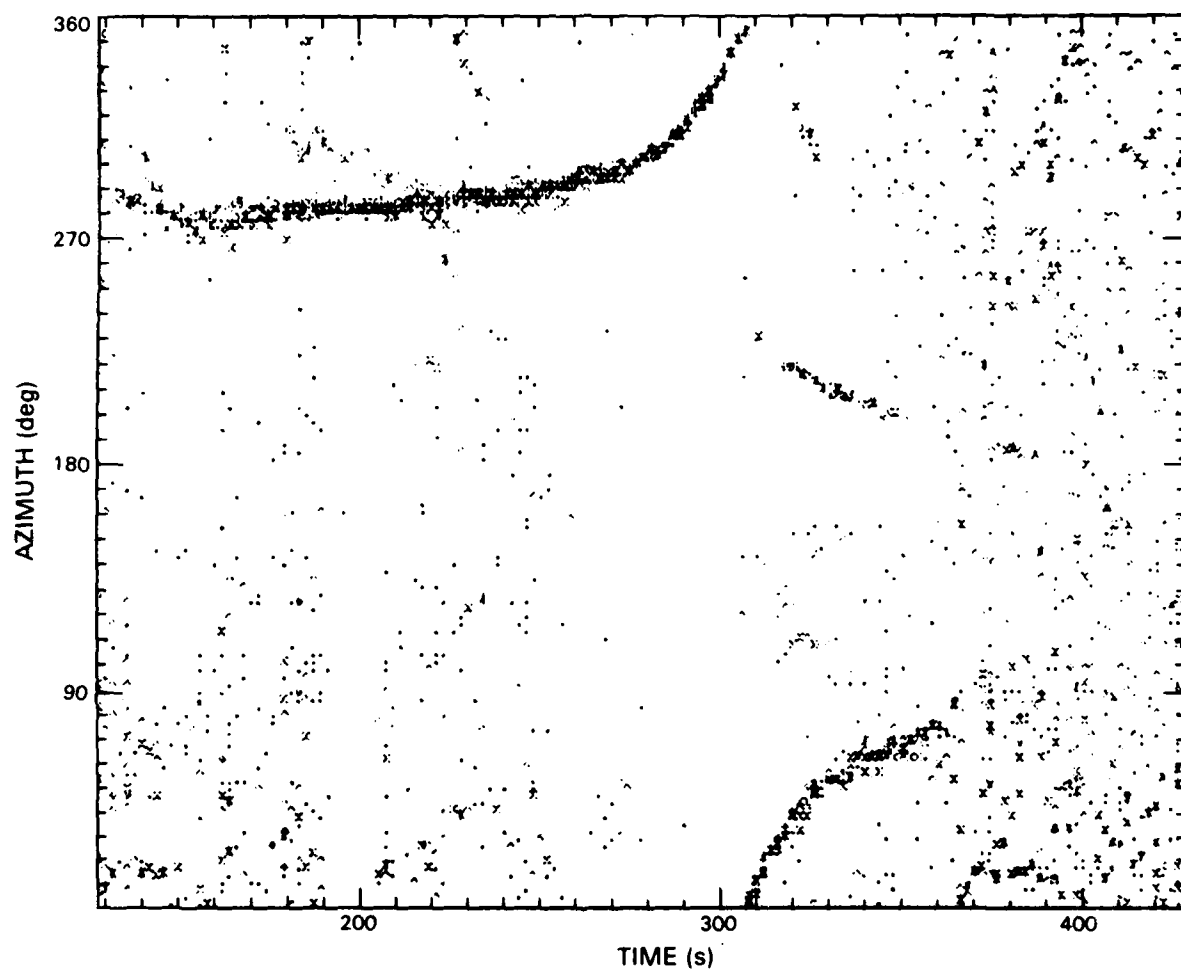
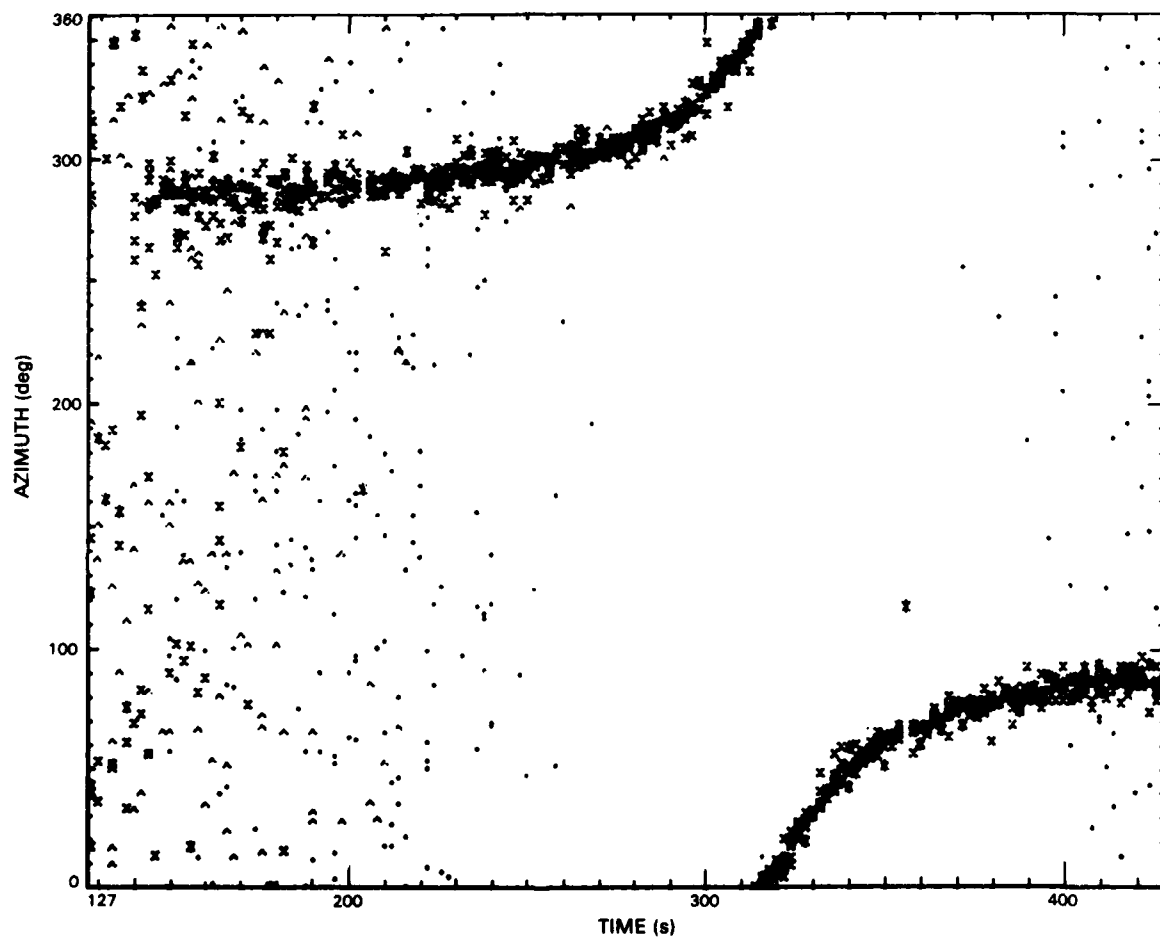Fig. II-9. True signal processor output for node J.

115684-R-01

18

Fig. II-10. Synthetic signal processor output for node J.

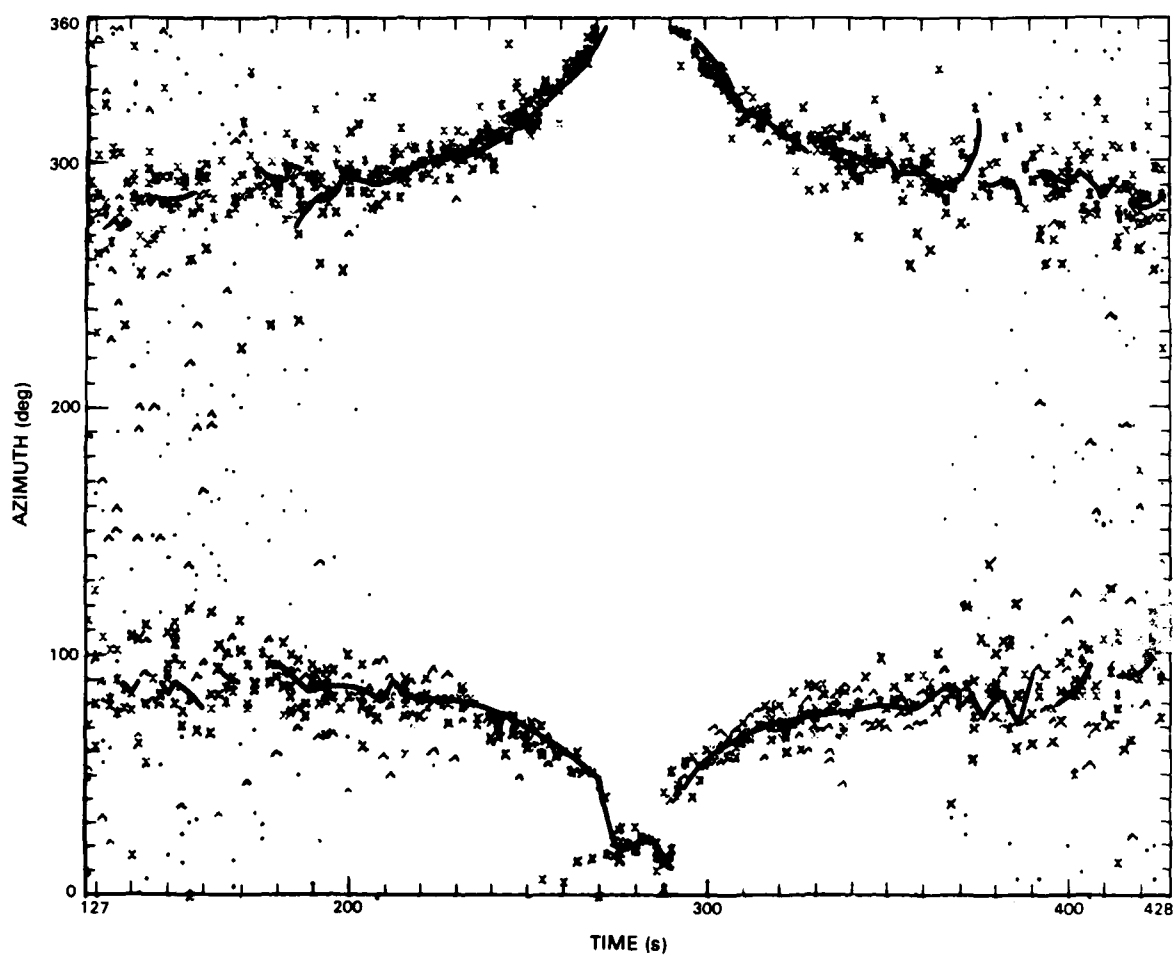Fig. II-11. Test-bed node locations and UH-1 helicopter flight paths for simulated experiment.

20

108158-R-03
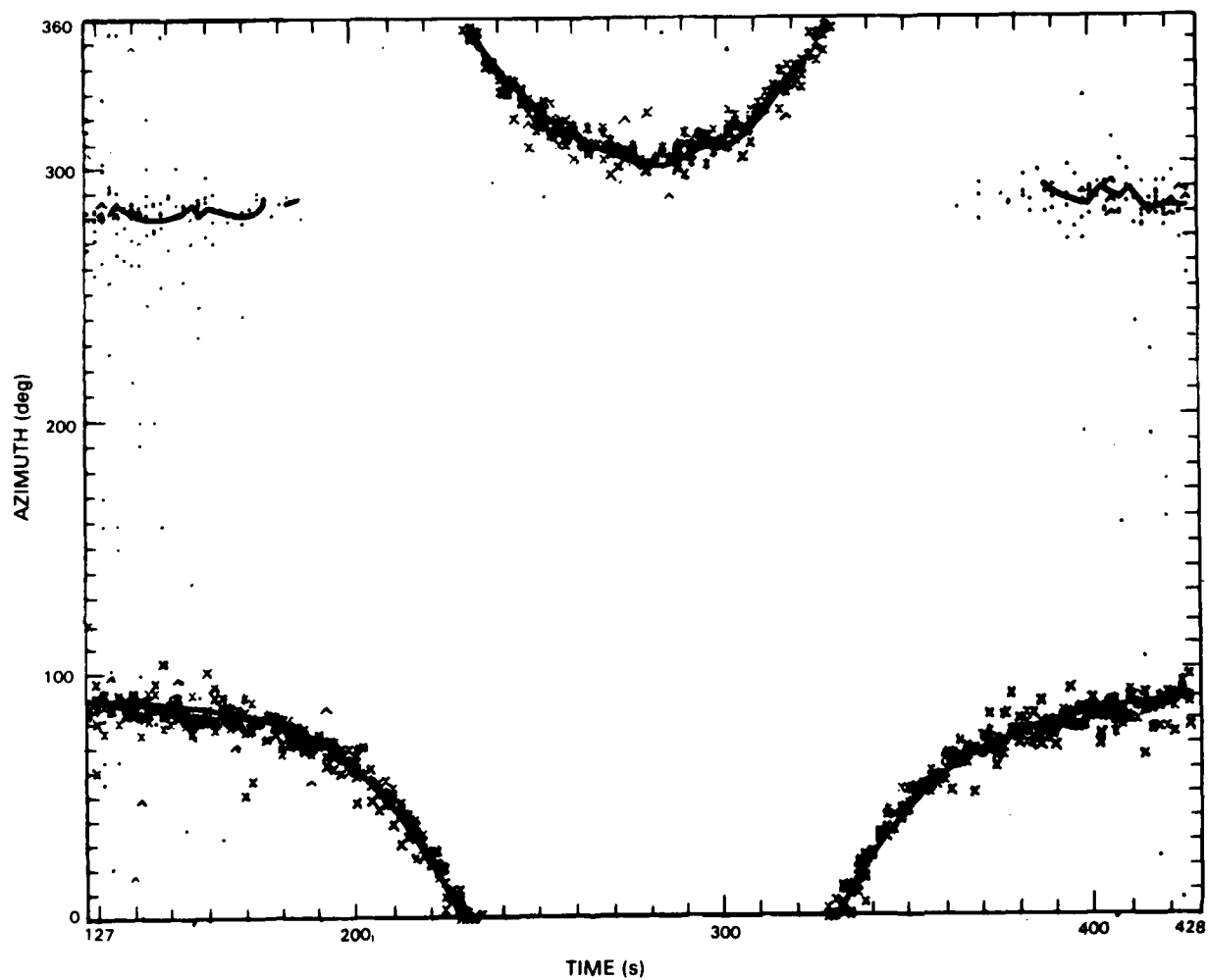
Fig. II-12. Synthetic signal processor output for node H.

Fig. II-13. Synthetic signal processor output for node L.

22

This explanation of the phenomena can be tested by generating synthetic detection data for which capture cannot occur because all frequencies are processed. Figure II-14 illustrates the result. In this case, the sensor and signal processor are also modeled as being noise free and of infinite resolution in angle. The pattern of detections contains no gaps; the detections are ideal. Although capture does not occur, the figure does demonstrate that even with ideal detections, the performance of the azimuth tracker is not perfect near the crossing point. The tracker clearly limits system performance over and above any limitations caused by less than ideal detection performance. Options being considered for tracker improvements include modification of the data association algorithms, changes to tracking filters, and scenario-based reinterpretation of low-level tracker outputs.

D. TRACK SEGMENT ASSOCIATION

The target tracking algorithms that now operate in the test-bed nodal processors produce target track segments using azimuth tracks from pairs of nodes. During this reporting period, we have developed and performed initial testing of simple algorithms for combining such track segments into composite tracks. Although the algorithms are simple, they are general in that the track segments could be from a diversity of sensor systems such as radar and IR as well as acoustics. These algorithms are candidates for future refinement and real-time implementation. Although they are simple and could be improved in many ways, they may serve the need for test-bed algorithms that associate and combine multiple node data for a single target into a single track to provide an integrated system view of the target.

Figure II-15 shows the pairwise track segments generated by three nodes for a straight line west-to-east pass of a UH-1 helicopter. Overlaid on the figure is a single straight line track that was the final output produced by the segment combining algorithms. These algorithms identify and remove extraneous track segments, replace curved segments with straight line approximations, and combine segments into composite tracks. Since the algorithm models tracks as a sequence of straight line segments and the actual track of the helicopter was a straight line, it is not surprising that the final track is a single straight line segment.

The following is an outline of the algorithms. The first step is to process the input segments and replace them with longer straight line constant velocity approximations if that is reasonable. This is done by recursively estimating the best straight line fit to each segment as time moves forward and starting a new straight line segment when the heading determined by the straight line from the end of that line to the next data point differs markedly from the already estimated heading. The resulting straight line segments are pruned by discarding very short segments and segments with velocity >100 m/s or <10 m/s. The resulting segments are input to the segment association and combining part of the process.

Two segments are associated based upon comparisons of track parameters corresponding to endpoints as shown in Fig. II-16. Given endpoints A and B, the points A' and B' corresponding to the same times as A and B, respectively, are determined. If track parameters match sufficiently well at A and A' and at B and B', then the segments are combined. The match depends upon heading, speed, and position. In each dimension the matching criteria is broad (within 45° in heading, within 50 m/s in speed, and within 500 m in location) to
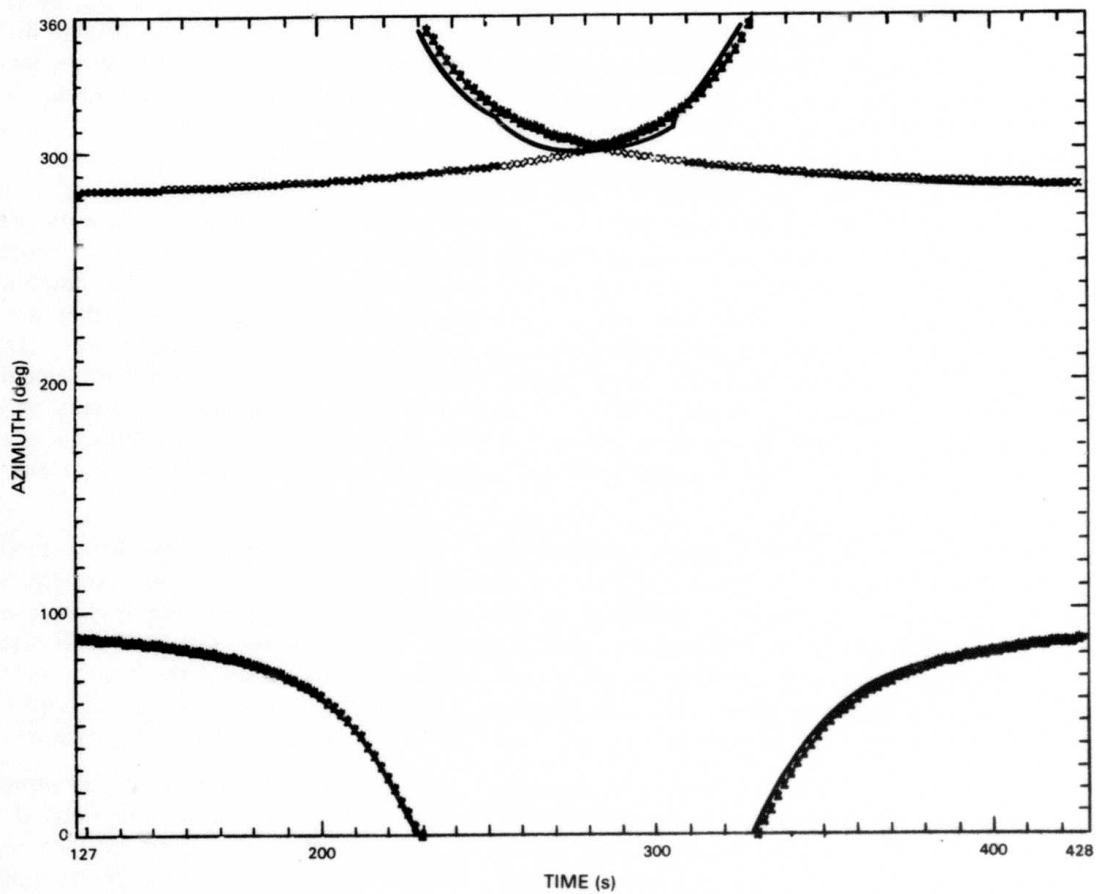
Fig. II-14. Ideal detections for node L.

24
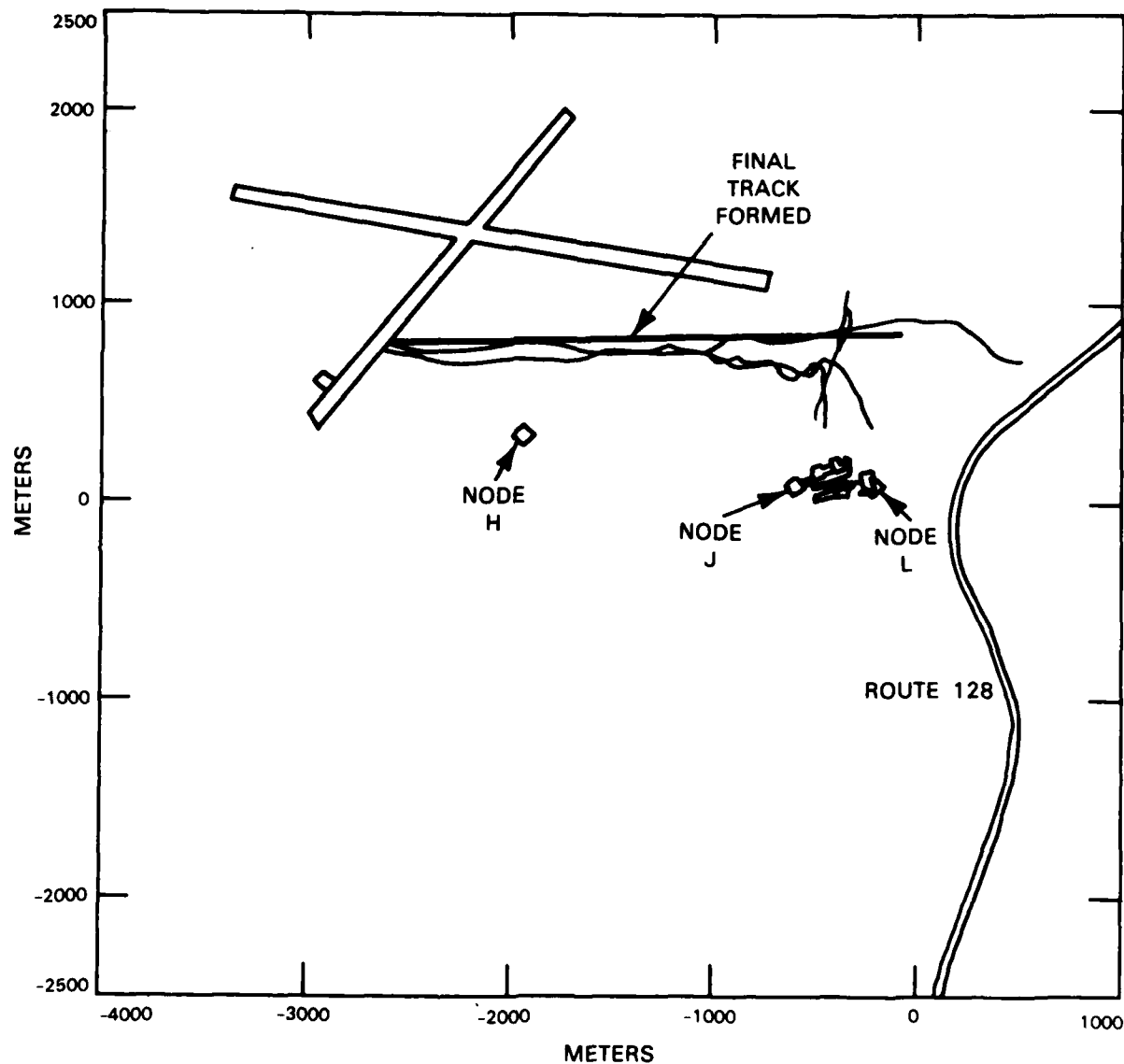
Fig. II-15. Comparison of input data to a final track formed by the association and segment combining process.

25

avoid rejecting correct associations; but the use of multiple dimensions results in a low false association rate. Overlapping and associated segments such as those shown in Fig. II-16 are combined to produce three abutting segments; one for the overlap portion and one for each of the ends. The association and combining is done recursively until there are no more segments that associate.

At that point, segments that have never been associated and combined with other segments are discarded and further heuristic pruning is done on the basis of target velocity, etc. The final stage is to process the resulting tracks that are composed of many short segments to approximate the trajectories by a small number of straight line segments if possible. This last stage is similar to the first stage applied to the original input tracks.

INPUT SEGMENTS
RESULTANT SEGMENTS
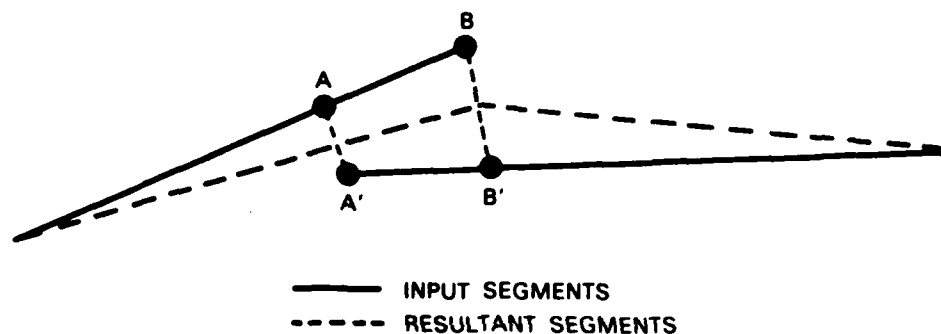
Fig. II-16. Track association testing and geometrical combination.

26

III. STANDARD MULTIPROCESSOR COMMUNICATING NODE DEVELOPMENT

A. NODAL HARDWARE

As reported in the previous SATS,* a prototype of a new standard nodal configuration is being developed. The general form of the new configuration is shown in Fig. III-1. The central element is a multiple microcomputer system that uses a standard Multibus as a system bus. A prototype of the Multibus system has been completed and is shown in Fig. III-2. This prototype system has now been connected to our PDP-11/70 developmental computer and to a nodal Signal-Processing Subsystem.

All elements of the prototype have been checked out and demonstrated. The serial and parallel communications links to the signal-processing system (SPS) have been tested. The hardware to reset the other processors in the node has been tested and demonstrated to function under software control. The Radio Unit Interface (RUI) board has been tested in loop-back mode and has been connected to the radio hardware that is being developed under the Communication Networks Technology (CNT) program. The RUI now is being used to aid in the debugging of the radio hardware.

The chassis for the prototype includes 12 edge connectors, the system bus, bus termination, and arbitration logic. The arbitration logic provides orderly access to the bus for the bus masters. An extension to the basic chassis accommodates power supplies, an I/O board, and a back panel. The I/O board provides for the orderly routing of signals and cabling and contains the logic for the front panel controls and for the Remote Reset Device (RRD). The RRD resets the Digital Communication Unit (DCU) when it receives a proper control sequence. The back panel provides for connections to the SPS, Radio Unit (RU) and various communication paths to local terminals, and the PDP-11/70 developmental computer.

The prototype chassis now contains three Stanford University Network (SUN) processor boards, one 512-kB shared memory board, a four-port serial interface board, a parallel port board with four 16-bit ports, and the RUI. One of the serial ports provides for exchange of commands and status information with the SPS and for down loading the SPS. The remainder are available for future use. One 16-bit parallel port with full handshaking is used to transfer data from the SPS. A second parallel port is used to provide outputs for front-panel displays and to provide programmed resets of nodal processors under control of the DCU. Two additional parallel ports are available for future expansion.

The RUI card is the only Multibus card in the system that has been custom designed and developed. A prototype RUI board has been debugged and is now operating in the prototype system.

Data transfer software was developed for the RUI in conjunction with hardware testing and debugging. The radio interface hardware contains an Intel 8089 microprocessor which

---

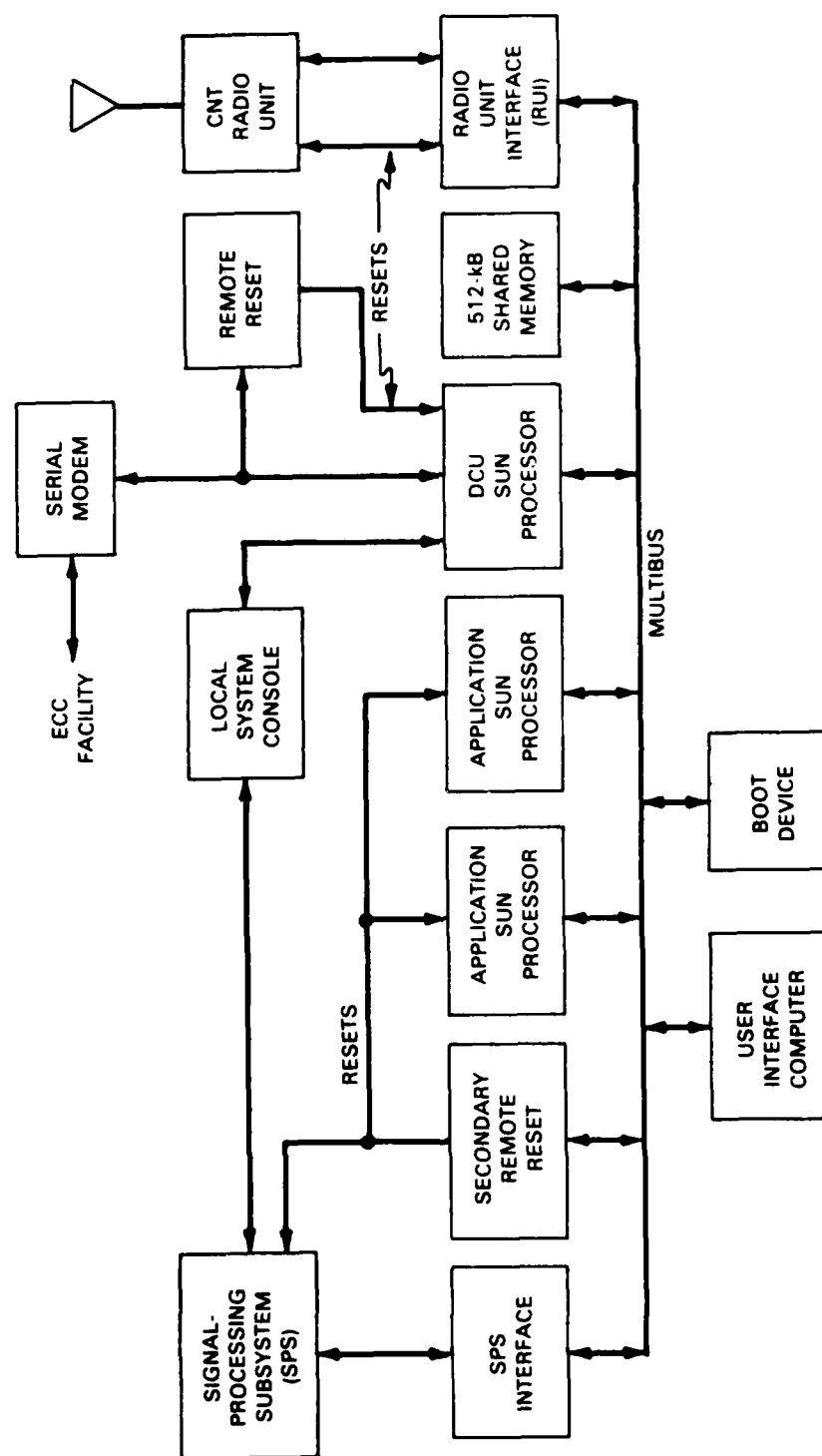*Distributed Sensor Networks Semiannual Technical Summary, Lincoln Laboratory, M.I.T. (30 September 1982).
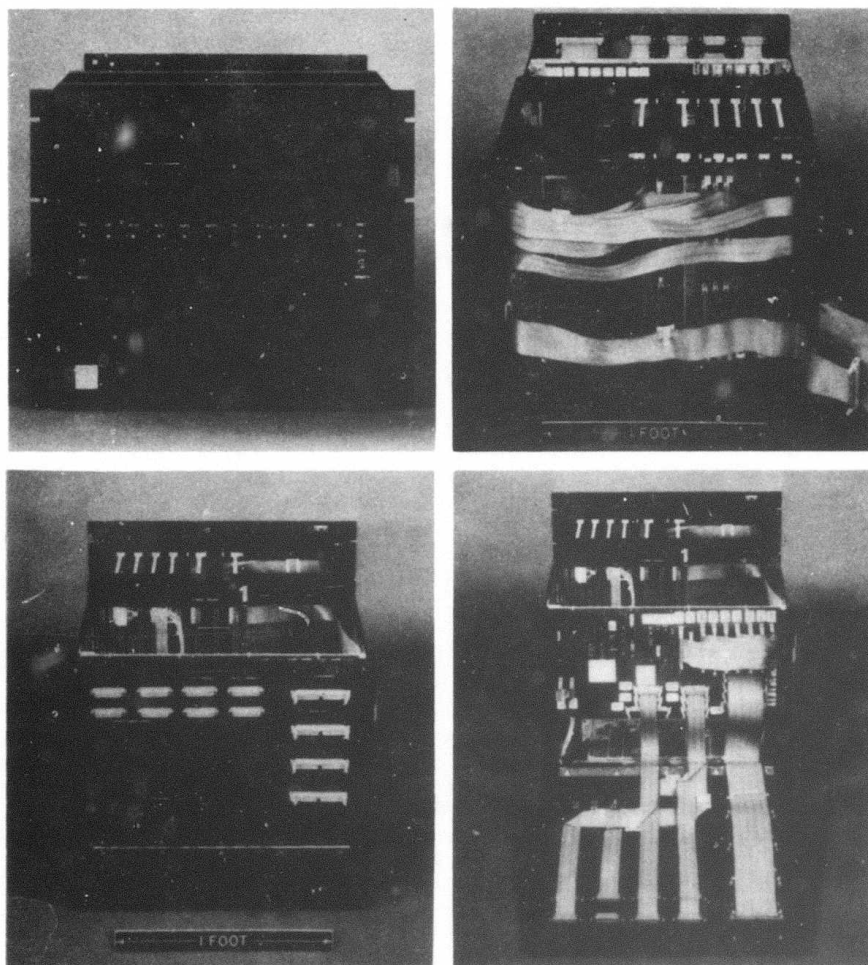
27

Fig. III-1. Standard test-bed node configuration.

28

126825-N-02

Fig. III-2. Photograph of the standard node prototype.

supports two independent DMA channels. The 8089 is used to perform DMA transfer of transmitted and received data between the radio and the MC68000 to increase system throughput. Channel software for the 8089 has been written and is operational. One channel handles the transmission of data to the radio and the other receives data from the radio.

The 8089 I/O processor is operated in its remote master configuration to provide high-efficiency operation. In this configuration it may be master of its local bus or the system bus. In a standard remote configuration the local bus is isolated from the system bus. We have included a System/Local Bus Interface (SLBI) as part of the RUI. The SLBI was implemented using four 20-pin chips for data and address transfers and two 14-pin chips for four local bus arbitration. This unit allows the DCU as well as the 8089 to access devices on the local bus, including the actual I/O logic that interfaces with the radio unit. The ability to directly use the DCU for diagnostic software and to develop capabilities before they are implemented for the 8089 has been of great utility.

Diagnostic programs that send user specified command packets and data to the radio have been written and tested. These programs are now being used by the radio developers to test and debug their design. This software transmits user defined command packets from the MC68000 across the interface to the radio. Radio clock slewing and transmission time selection have been demonstrated through this mechanism.

As part of the debugging process for the radio unit interface, a number of desirable improvements have been identified. Lack of space on the initial board has made it impossible to implement all the enhancements on the first prototype. A new prototype is now being designed on a board that will accommodate a higher chip density.

RUI enhancements include a provision to make control registers readable so that independent portions of the software can determine the status of the control registers, the addition of an extra status register, provision for additional interrupts and the addition of an Autonomous Reset Circuit (ARC). The additional interrupt was required to synchronize the reading of status information from the radio unit. The extra status register provides for handling this interrupt and provides for future expansion. The ARC provides for node specific resets by means of reset messages received by the radio unit. Provision has been made for several reset options, including reset for the case of a catastrophic software failure.

In support of the redesign activity an existing computer-aided design package, written in APL, has been modified to accept a MUPAC board configuration.

B. SYSTEM SOFTWARE

During this reporting period, we have completed the design and partially implemented the system software for the new standard multiprocessor nodes. A major objective has been to implement software which will support our existing applications programs. Another important objective has been to lay a foundation for future work, including enhancements to applications programs and to systems capabilities in the distributed DSN environment.

The software design has been influenced by our experience with the previous interim nodes. In particular, we have continued to provide UNIX-like system calls and to code almost exclusively in C. A different set of low-level synchronization primitives has been provided, partly because of the nature of the new hardware, but also to permit preemptive

process scheduling. Finally, considerably more attention has been paid to providing a strict hierarchy of functionality in the system design.

The differences in hardware between the new and old nodal configurations has had an effect on the software design. For example, the coupling of processors through a shared memory required the development of a new set of low-level synchronization primitives. Perhaps more obviously, the presence of a different set of peripherals on the SUN-based nodes required a new set of device drivers. Although the memory mapping facilities of the SUN processors are not used in the present design, their availability has influenced a number of decisions with the idea that they eventually will be utilized.

The system software is organized into three sections. At the lowest level, the kernel provides a set of basic services, including facilities for process creation, memory allocation, and interprocess synchronization. At a slightly higher level, the device-handlers perform physical I/O and, finally, there is a group of subroutines and processes which define the user interface.

The kernel is a set of subroutines which are shared by all processes in the node. These subroutines execute in supervisor state regardless of the state of the process which called them. Services provided by the kernel fall into three categories. Process creation and definition are performed by a set of calls similar in function to "fork" and "exec" in UNIX. Dynamic memory allocation is available through a set of calls which are similar to the UNIX subroutines "malloc" and "free." Interprocess synchronization is provided in the form of semaphores and interprocess queues.

The first level up from the kernel consists of a set of processes which perform physical I/O. These processes run in a privileged state and have direct access to nodal hardware. Their role is similar to that of interrupt service routines in many operating systems.

The final layer of the run-time system is composed of a set of processes and shared procedures which provide the user interface. The user interface includes, for example, routines for performing logical I/O (i.e., read and write). The processes which compose this layer are not privileged and do not have direct access to nodal hardware.

At this writing, the kernel has been completed and is running in a SUN processor. Various parts of the user-interface layer have been completed, including the capabilities for naming logical devices.

C. RADIO COMMUNICATION PROTOCOLS

The essential near-term radio communication requirements for the DSN test bed include:

(1) Local broadcast
(2) Range measurement
(3) Performance measurement capabilities.

The basic protocols have been designed and software is now under development. The protocol includes listen-before-talk techniques to reduce the number of lost broadcast messages. It provides for multiple broadcast channels with a guaranteed minimum fraction of the available bandwidth associated with each channel. It provides for discarding old messages if necessary to reduce the delay in broadcasting new information and it provides a mechanism for nodes to enter and leave the network.

31

CHANNEL 1 ... CHANNEL N RANGING QUEUE

HIGH PRIORITY

FIFO QUEUE OF BROADCAST MESSAGES
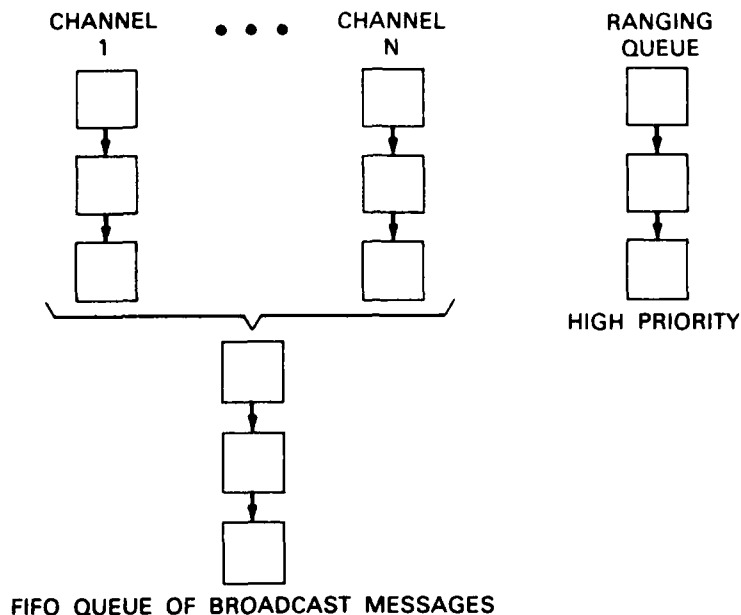
Fig. III-3. Message scheduling queues.

The following summarizes the features of the implementation.

1. Message Scheduling

Messages from the application layer are divided into groups called channels. The assignment of messages to channels is user defined. The design provides for each channel to have a priority but in the initial implementation the messages from all user channels will be merged into a single FIFO queue. A priority scheme for choosing messages from the channels will be implemented later. Ranging messages will occupy a separate queue and are given highest priority (see Fig. III-3). Messages on the ranging queue will be scheduled for transmission before any other messages.

Once a message is selected from a queue, it is scheduled for transmission with probability p for the next available 10-ms slot. The transmission time will be chosen at random within a slot. The radio uses a listen-before-talk protocol and will abort a transmission if a reception is already in progress at the scheduled transmission time. Aborted transmissions are rescheduled with probability p for the next available slot. Subsequent retransmissions will decrease the value of p.

Each message will have a time to live associated with it and will be discarded if that time expires before the transmission takes place.

32

2. Ranging

Ranging messages will be used to calculate distances between nodes and to obtain clock differences between nodes. The protocol will utilize three messages although only two are actually required to obtain an internodal range measurement and clock difference.

The protocol involves interactions between a pair of nodes as follows. A ranging request message is sent from node A to B. This message contains A's time of transmission. Node B responds by preparing and broadcasting a ranging reply packet. The ranging reply packet will contain B's measurement of the node B arrival time for the ranging request from A, node B's time of transmission of the ranging request response, and a copy of the ranging request packet received from A. Upon receipt of the ranging reply message from B, node A formulates and broadcasts a range acknowledgment packet. The range acknowledgment packet contains A's measurement of the time of arrival of the range reply packet from B plus a copy of the range reply packet from B.

The last of the three transmissions in the ranging protocol distributes information from node A to B and other nodes. Only the first two messages are needed to calculate range and clock offset. The formulas for the calculation of transit time between nodes and clock offset are:

$$\text{Transit time} = [(\text{TOA node B} - \text{TOT node A}) + (\text{TOA node A} - \text{TOT node B})]/2 \text{ and}$$

$$\text{Clock difference} = \text{TOA node B} - \text{TOT node A} - \text{Transit time}$$

where TOA and TOT denote time of arrival and time of transmission. The range is calculated from the transit time.

The protocol described above is based upon the incoherent mode of time-of-arrival measurement that will be provided by the radio units. A separate more complex protocol will eventually be required to make use of the coherent mode that will also be provided by the radio units. The primary reason for this is that coherent time-of-arrival measurement requires special packets with known contents and additional information must be exchanged using separate packets.

When a ranging request is received, its reply will go out as soon as possible. Ranging messages will be the highest priority messages in the system. The limit on the number of retries is zero for the reply but not for the request. This policy for replies minimizes the complications resulting from the fact that code slots are only nominally 10 ms in length and might be changed from slot to slot.

3. Addressing

The length of node addresses is 16 bits. This address length was chosen to be consistent with the packet radio CAP protocol. Source and destination addresses will be specified in all messages. The destination address FFFF indicates a broadcast message that is intended for all nodes that receive the message. Broadcast messages are not forwarded. Addressing broadcast messages to a specific neighboring node or a subset of neighboring nodes will be performed by including a list of destination node addresses in the message header.

4. Statistics

Communication statistics will be collected by each node. The table containing the statistics will be automatically and periodically reinitialized to prevent overflow. It will also be possible to remotely command a node to transmit the contents of the table and to reinitialize the table.

The statistics collected by each radio will include: (a) number of broadcast messages received by the node, (b) number of received broadcast messages specifically addressed to the node, (c) number of broadcast messages transmitted by this node, (d) number of ranging messages received, (e) number of ranging messages transmitted, (f) number of messages transmitted for each channel, (g) number of messages discarded due to age, (h) number of transmissions aborted due to ongoing receptions, (i) number of messages with checksum errors, and (j) total number of words transmitted. The count of the number of broadcast messages received by the node will be broken down according to the node from which the messages are received.

IV. MISCELLANEOUS ACTIVITIES

A. SELF-LOCATION ALGORITHMS AND SOFTWARE

We have been working with the Columbia University DSN research group to restructure, install, and operate their position location software on the DSN test bed. The initial objective is to establish an overall software structure, software standards, and software exchange mechanisms to support continuing cooperative development of and experimentation with self-location algorithms. Earlier interactions with the Columbia group established general software compatibility through the use of the C language.

Three software modules were obtained from Columbia: (1) an input generation and display program, (2) a position location program, and (3) an output display program. Figure IV-1 shows how this software was integrated into the test bed.

Modules (1) and (3) were combined into a position location graphics support process which runs in conjunction with the User Interface Program (UIP) on the Experiment Control and Communication (ECC) computer. This process is controllable from the UIP user terminal. It can be used to produce an edge distance data file from a graph file. The graph file is a file giving the names and locations of a set of nodes and the edge file is essentially a set of simulated range measurements. The graphics support process can also be used to plot the input graph and edges.

The UIP can be used to send the edge data to the remote position location software in the form of command messages. The position location module runs as an application process in a remote MC68000 processor. It calculates node positions from the edge data and returns results to the UIP in the form of messages. The UIP spools these to an output file which can then be examined and displayed using the graphics support software on the ECC.

Position location involves matrix inversion and other mathematical operations in which normalization is essential to prevent arithmetic over and under flow. Rather than impose unnecessary constraints or complexity upon the algorithm development activity, we have provided a floating-point software library for the MC68000 systems. This library is real-time in that it handles errors without aborting the arithmetic operation and returns reasonable values under all circumstances. This library includes routines for double precision add, subtract, multiply, divide, and exponentiation. It is written in "C" and 68000 assembly language and is reentrant so that it can be used by multiple tasks within a nodal processor.

All the code discussed above has been checked out and found to operate correctly with a four-node graph as input. To investigate larger graphs, the static memory allocation scheme used by Columbia must be changed so that a single MC68000 without disk-based virtual memory management can handle larger networks. We are in the process of changing the software from a static to a dynamic memory allocation scheme that will use the available memory more efficiently and will allow us to operate with larger network configurations. When this is completed the revised software and associated documentation will be provided to Columbia to serve as the basis for their future software development.
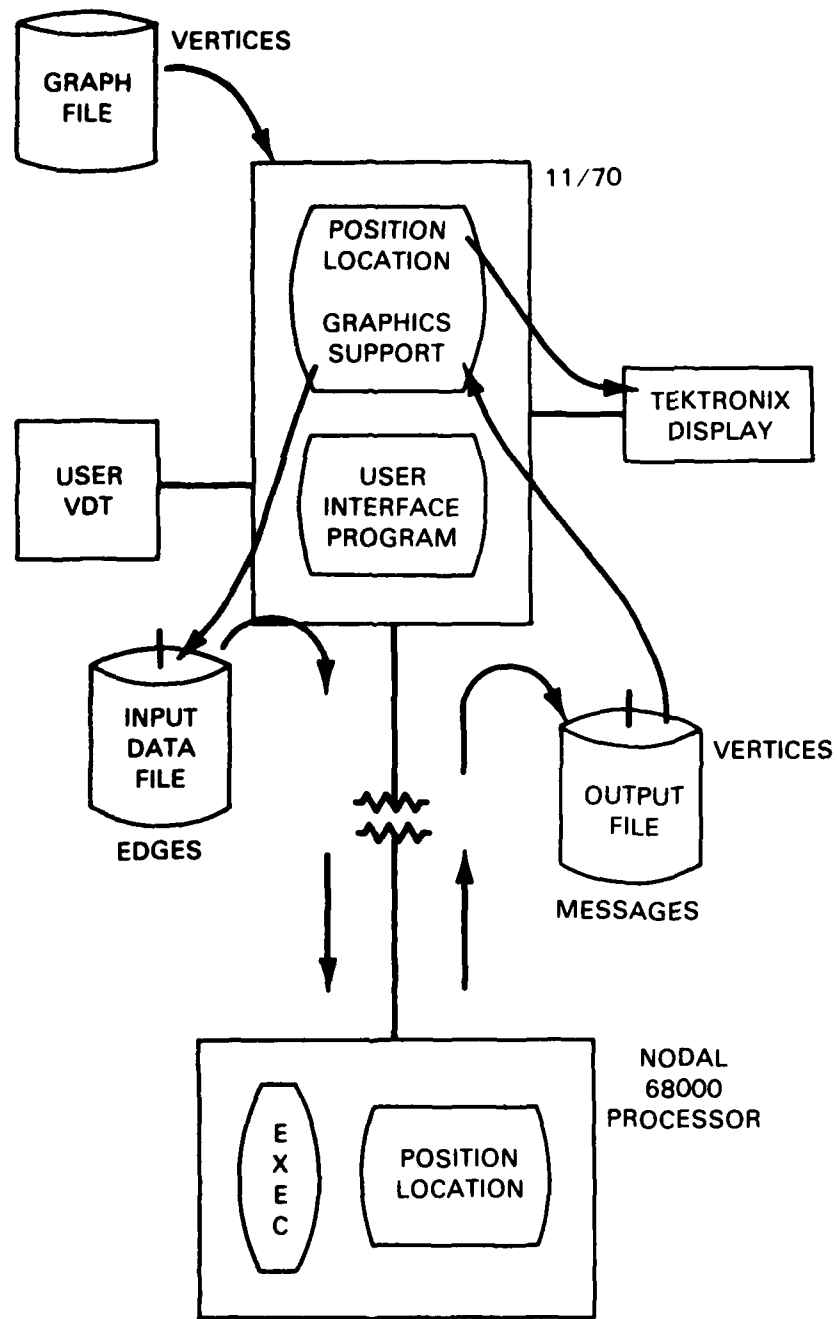
Fig. IV-1. Integration of the position location software into the test bed.

129706-N

36

B. USER INTERFACE COMPUTER

An order has been placed for a MC68000-based UNIX system which will be used as a prototype for a test-bed User Interface Computer (UIC). This prototype system will support the UIP (currently running on the PDP-11/70 computer) and will provide both user interface and communications emulation services for the test bed. The prototype is being procured with standard tape drives and a large disk system. In addition to serving as a UIC prototype it will replace the PDP-11/70 as the ECC computer and provide additional test-bed software development resources.

The prototype UIC system includes a high-resolution bit map display that can be used to provide more dynamic and useful graphic displays than are possible with the storage tube displays now employed in the test bed. The new displays provide hardware vector generation and DMA memory to bit map data interchanges.

The prototype is also being procured with ETHERNET interface hardware and with the UNET software package* that will provide IP/TCP communication support for the ETHERNET. This will be used to provide FTP, VTP, and mail service between the MC68000 UNIX system and the PDP-11/70 and, indirectly, to the ARPANET.

As noted above, the prototype UIC will also be used as the test-bed ECC and will be connected to the remote nodes through land lines. A later version, with fewer and lower capacity peripherals, will be directly attached to one of the test-bed nodes.

C. ARPANET SOFTWARE

During this reporting period, the ARPANET converted to the new standard IP/TCP protocols. UNET software, with ARPANET enhancements, was obtained and installed on the PDP-11 that serves as the 11-xn host on the network, the test-bed ECC, and our primary research computer.

D. SIGNAL-PROCESSING RESEARCH TOOLS

In an effort to evaluate the signal-processing options for DSN, it is necessary to examine the signal-processing output in a variety of ways. Since the output information has several dimensions (time, frequency, azimuth, elevation, signal level, signal-to-noise ratio), it is very difficult to represent this information in a single plot. Up to now, we have mainly utilized azimuth vs time plots for the signal level. We have now developed several other types of plots — analysis frequency vs time, number of peaks vs time, total signal level vs time, azimuth vs frequency, elevation vs time for signal level, etc.

We have restructured the signal-processing software package that is used for evaluation of signal-processing strategies. This package uses much of the signal-processing code that has

---

*M. Bonnain, B. Borden, and G. Shaw, "UNET Networking Software Reference Manual," 3-COM Corporation, Mountain-View, California (9 February 1983).

37

been developed in the past few years. However, the overall organization has been changed to give the user more flexibility in changing parts of the package. The key to this flexibility lies in its modular nature.

The restructured package consists of independent programs that take an input file, process it in a particular manner, and produce an output file. Currently, the package contains an FFT program, a covariance estimation program, a covariance inversion program, and an MLM program. Additionally, there are plotting programs that plot the output files of each of the above programs. The control programs are UNIX shell programs. This allows us to run each individual program of the package with all the user's current memory allocations devoted to that program. Thus, for example, the FFT program can compute much longer FFTs than programs which perform other functions in addition to the FFT.

E. DISTRIBUTED DETECTION AND ESTIMATION

A distributed detection and estimation session of the American Control Conference will be held 22-24 June 1983. A paper* has been prepared for presentation at that meeting and will be included in the conference proceedings.

---

*J.R. Delaney, R.T. Lacoss, and P.E. Green, "Distributed Estimation in the MIT/LL DSN Test Bed."

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ESD-TR-83-028	**2. GOVT ACCESSION NO.** AD-A133 250	**3. RECIPIENT'S CATALOG NUMBER**
4. TITLE *(and Subtitle)* Distributed Sensor Networks		**5. TYPE OF REPORT & PERIOD COVERED** Semiannual Technical Summary 1 October 1982 — 31 March 1983
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Richard T. Lacoss		**8. CONTRACT OR GRANT NUMBER(s)** F19628-80-C-0002
9. PERFORMING ORGANIZATION NAME AND ADDRESS Lincoln Laboratory, M.I.T. P.O. Box 73 Lexington, MA 02173-0073		**10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS** ARPA Order 3345 Program Element Nos. 61101E and 62708E Project Nos. 3D30 and 3T10
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, VA 22209		**12. REPORT DATE** 31 March 1983
		13. NUMBER OF PAGES 46
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)* Electronic Systems Division Hanscom AFB, MA 01731		**15. SECURITY CLASS.** *(of this report)* Unclassified
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

multiple-sensor surveillance system	acoustic sensors
multisite detection	low-flying aircraft
target surveillance and tracking	acoustic array processing
communication network	digital radio

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This report describes the work performed on the DARPA Distributed Sensor Networks Program at Lincoln Laboratory during the period 1 October 1982 through 31 March 1983.

DD FORM 1473 1 Jan 73	EDITION OF 1 NOV 65 IS OBSOLETE	